AD-A093 216 GENERAL ELECTRIC CO PITTSFIELD MA ORDNANCE SYSTEMS F/G 9/2
ELECTRICAL CHARACTERIZATION OF MICROPROCESSORS AND THEIR SUPPOR--ETC(U)
OCT 80 B W HAJDUK, T M OSTROWSKI, D TUCKER F30602-78-C-0172

UNCLASSIFIED RADC-TR-80-325 NL

END
DATE FILMED
1-81
DTIC

ELECTRICAL CHARACTERIZATION OF MICROPROCESSORS AND THEIR SUPPORT CHIPS

General Electric

R. W. Hejduk
T. Gorowski
D. Tasker, et al

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

[redacted]

[redacted], Colonel, USAF
Deputy, Installation & Logistics Division

FOR THE COMMANDER:

[redacted], USAF
Acting Chief, Plans Office

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: RADC-TR-80-325	2. GOVT ACCESSION NO.: AD-A093 216
	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): ELECTRICAL CHARACTERIZATION OF MICROPROCESSORS AND THEIR SUPPORT CHIPS.	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report. Apr 78 — May 80
	6. PERFORMING ORG. REPORT NUMBER: N/A
7. AUTHOR(s): W. Hajduk, J. Ostrowski, L. Tucker, et al	8. CONTRACT OR GRANT NUMBER(s): F30602-78-C-0172
9. PERFORMING ORGANIZATION NAME AND ADDRESS: General Electric/Ordnance Systems, 100 Plastics Ave, Pittsfield MA 01201	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62702F, 23380150
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (RBRA), Griffiss AFB NY 13441	12. REPORT DATE: October 1980
	13. NUMBER OF PAGES: 90
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Same	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Regis C. Hilow (RBRA)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microprocessors
support devices
MIL-M-38510 slash sheets
functional tests

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The objective of this effort was to develop MIL-M-38510 slash sheets for microprocessors and their support devices. The report includes the technical analyses of manufacturer generated functional test programs and the changes made to them. Four slash sheets (not included in this report) were developed. They are /460 for the 9900A, /443 for the 2909, /445 for the 2910, and /401 for the 6821. User and manufacturer comments to the /440 and /421, developed under a previous contract (Report RADC-TR-

(Cont'd)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD A093175

Item 20 (Cont'd)

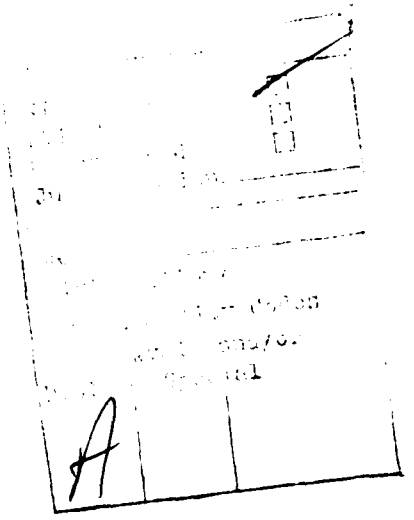
78-138), were reviewed with the manufacturers and changes were made to the slash sheets were required.

PREFACE

This Final Report was prepared by General Electric Ordnance Systems, 100 Plastics Avenue, Pittsfield, Massachusetts, for Rome Air Development Center, Griffiss Air Force Base, New York, under contract F30602-78-C-0172.

It covers the period from April 1978 to May 1980. Mr. Regis C. Hilow, RBRM, was the RADC Project Engineer.

The work on this project was performed by the Electronic Circuits Engineering Operation and Components Engineering Unit. Project responsibility was held by Messrs. Thomas M. Ostrowski and Barney W. Hajduk of Circuit Design Engineering. Key individuals who made significant contributions to this report were Messrs. Donald Tucker, John Luzzi, Lawrence Deluca, James Schwehr, and Richard English.



CONTENTS

LIST OF FIGURES

LIST OF TABLES

Evaluation

The prime objective of this effort was to evaluate the electrical design and to electrically characterize complex LSI microcircuits to assure their performance in a military environment. Emphasis was placed on completing the electrical characterization of the 2900 family and to initiate activity on the support devices for the 6800 microprocessor.

The methodology for development of the electrical test requirements for these parts did not differ from that used on other parts with similar complexities and process technologies such as the 8080A. Each device was partitioned into functional logic blocks such as registers, counters, etc. and all data and control paths identified. Test vectors were developed to test each block and to guarantee the integrity of each control and data path. The complete instruction set was tested under various conditions and the effects of pattern sensitivity evaluated. Throughout this activity emphasis is placed on assuring that the final test program can be implemented by the device vendors on their automatic tester and that total test time is optimized.

By the completion of this effort the 9900A microprocessor was completely characterized and coordinated with DOD/NASA agencies. Military specifications were also prepared for the 2909, 2910 and 6821 in the course of this contract. DOD/NASA coordination is near completion for these specifications. Other special topics such as the use of dedicated testers by vendors are also addressed in this report.

RADC, as preparing activity of MIL-M-38510, General Specification for Microcircuits, is responsible for managing the development, preparation and coordination of detail slash sheets for this specification. This study and future studies will be directed at providing on timely basis DOD/NASA system builder state-of-the-art VLSI devices that satisfy military environmental requirements.

REGIS HILOW
Project Engineer

SECTION I

SUMMARY

This report details the functional, AC, and DC test analysis performed on three microprocessor support devices and the follow-on test development performed on a previously characterized microprocessor and support device. The results of the support device analysis were used in the development of MIL-M-38510 slash sheets (not included in this report). The characterized devices are listed below.

The "Procedure for LSI Functional Test Development", which was established on previous RADC contracts and documented in Report RADC-TR-78-138, June 1978, formed the basis for this evaluation. The same report also included the results of the characterizations of the 2901A Four Bit Microprocessor Slice and the 8212 Eight Bit I/O Port which were followed-up on this contract.

The capabilities of a dedicated benchtop tester (Megatest Q8000) were reviewed to determine it's applicability to MIL-M-38510 testing.

Characterized Devices

9900A - 16 Bit I^2L Microprocessor

6821 - Peripheral Interface Adapter

2909 - Microprogram Sequencer

2910 - Microprogram Controller

SECTION II

INTRODUCTION

This characterization was an extension of similar efforts performed for other microprocessors and support devices on previous RADC contracts. On these earlier contracts a test philosophy was developed for microprocessors and other LSI devices. This philosophy was subsequently used to develop functional tests for inclusion in the first MIL-M-38510 slash sheets for microprocessors. Various support devices for microprocessors were also characterized on these contracts and since the support devices were not as complex as the microprocessors, it was possible to perform gate level analysis on them during functional test development.

This report includes the evaluation of the functional tests provided by the vendors for the 9900A 16 Bit I^2L Microprocessor(/460), the 6821 Peripheral Interface Adapter (/401), the 2909 Microprogram Sequencer (/443), and the 2910 Microprogram Controller (/445). A functional block level analysis was performed on the 9900 and the 6821 using the test philosophy developed on the previous contracts. The vendor for the 2909 and 2910 provided gate level logic diagrams which were used for the analysis of these devices.

During this characterization the vendors for the 2901A Four Bit Microprocessor Slice and the 8212 Eight Bit I/O Port began efforts for qualification. The vendor for the 2901 A reviewed slash sheet 440 and began writing test programs to conform to its requirements. Some test difficulties were encountered during this effort and it became necessary to make changes to the specification. Section VII of this report describes the problems encountered and their solutions.

The 8212 vendor submitted his in-place tests for use in lieu of the tests in the 421 slash sheet. These were reviewed and comments and recommendations were forwarded to RADC, DESC, and the vendor. The vendor is presently reviewing them so that a disposition of the testing can be made. Section VIII describes the review of the 8212.

As more vendors began to use dedicated benchtop testers, it became necessary to learn their capabilities as a means of determining their applicability to MIL-M-38510 testing. Section IX describes GE's preliminary review of the Megatest Q8000.

SECTION III

FUNCTIONAL TEST EVALUATION FOR THE 9900A

Objective

The purpose of this evaluation was to review the functional test which Vendor F had submitted for inclusion in the slash sheet for the 9900A. The method used has been previously documented in RADC Report RADC-TR-78-138 and will not be repeated here. In this report, it will be referred to as the checklist for LSI testing.

Summary

The evaluation of the functional test for the 9900A was different from that for other microprocessors in that more of it was performed by the vendor. On other devices, the vectors were evaluated by GE and the vendor was asked questions when clarification was required. The technical contact for the 9900A had been involved in the development of the functional test and it was easier for him to evaluate some parts of the test.

Vendor F supplied the following information for the evaluation.

1. A hexadecimal listing of their test vectors.

2. The output (FUSIM listing) from their functional simulator program. This listed the status of the internal registers and other selected points for each vector.

3. An assembly listing of the functional test.

4. A more detailed block diagram than the one in the catalog. This block diagram is shown in Figure 3-1.

5. A description of the machine cycles required to execute each of the op codes.

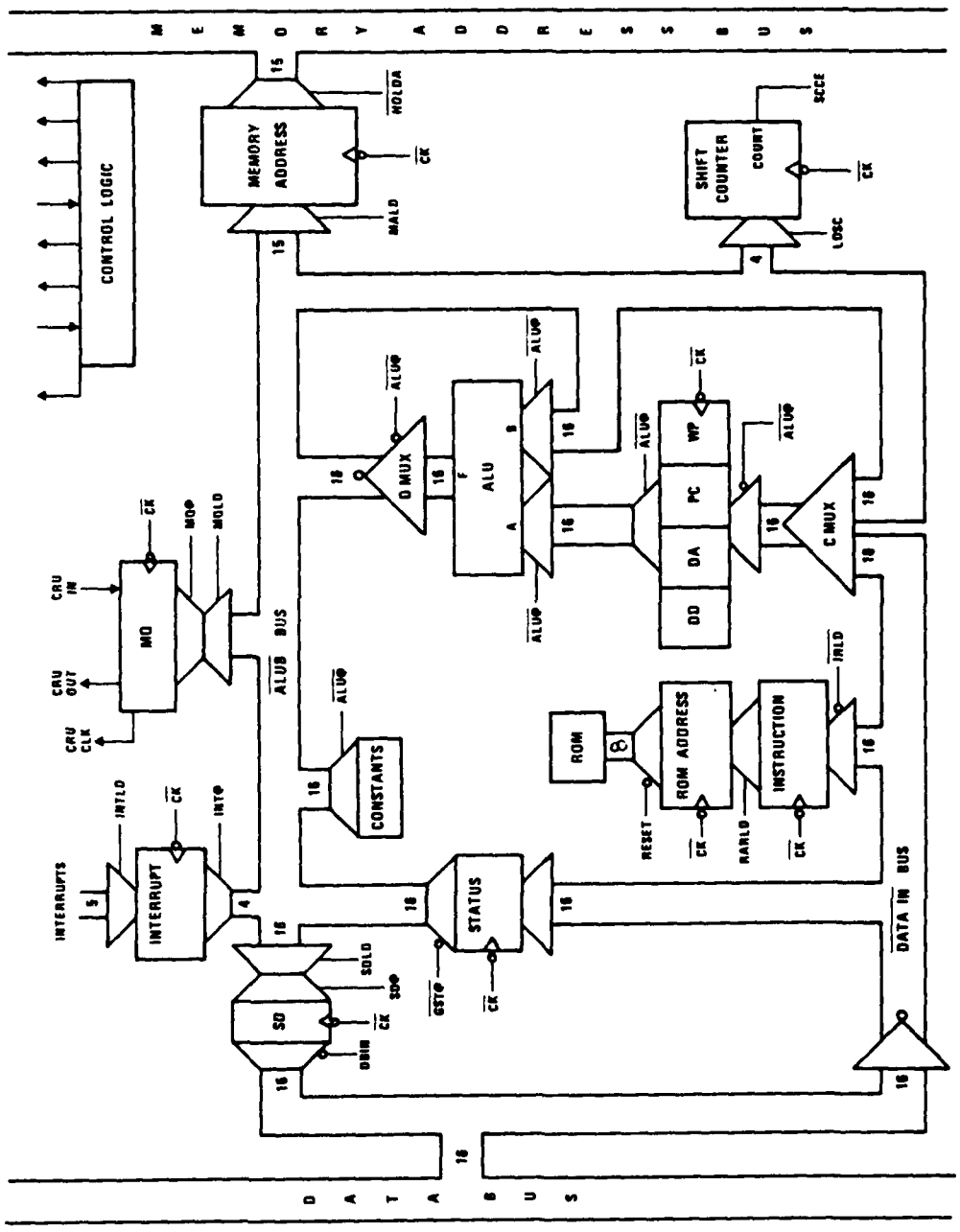


FIGURE 3-1 9900 BLOCK DIAGRAM

This information was reviewed and a list of tests required to check the 9900A was developed. This list, which formed the basis for the evaluation, was sent to Vendor F along with a list of questions for them to answer. While Vendor F was investigating these questions, GE was evaluating other parts of the test.

Vendor F's answers were reviewed for completeness and in some cases, they were contacted for additional information. A meeting was held at the vendor's facility to review the entire evaluation and to obtain answers to any questions that remained.

During the evaluation it was found that not all of the required tests were performed. Vendor F agreed to make the necessary changes to their test. A summary of these changes is presented at the end of this report.

Discussion

The 9900A was divided into blocks or areas that had to be tested or verified and then, using the checklist for LSI testing as a guide, a list of tests required to check each of them was developed. The following areas were defined:

1. $\overline{\text{RESET}}$ verification
2. ALU operation
3. Conditional branch logic
4. Instruction register and decoding logic
5. Multiplexer verification
6. Status register
7. Interrupt circuitry
8. CRU interface
9. Register verification
10. Control functions
11. Data bus verification
12. Miscellaneous verification

The required tests and results of the evaluation for each of these areas are presented below:

1. $\overline{\text{RESET}}$ verification

   a) Verify that the proper memory addresses are generated for

the acquisition of a workspace pointer (WP) and program counter (PC) from memory.

b) Verify that the status register is reset properly. This requires that each bit be a one prior to reset.

c) Verify that $\overline{WE}$ and CRUCLK are inhibited.

During a reset cycle the contents of the WP, PC, and status register (ST) are stored and then these registers are reloaded, the latter with all zeroes. Although a reset was done at the beginning of each test block the outputs were not monitored because the contents of the internal registers were not known. Two resets were done after the registers were initialized. This verified that the registers were stored into memory properly and that the WP and PC were reloaded with the contents of memory addresses 0000 and 0002 respectively. However, the ST was not checked after the reset to verify that it contained all zeroes. Vendor F agreed to add a test which would verify that the ST was reloaded with all zeroes during reset.

The outputs of the 9900A were not monitored on every vector in the test. Vendor F agreed to change the test so that DBIN, $\overline{MEMEN}$, $\overline{WE}$, CRUCLK, HOLDA, WAIT, and $\overline{CYCEND}$ are monitored on every vector. With this change it is verified that $\overline{WE}$ and CRUCLK are inhibited during reset.

2. ALU Operation

a) Verify add and subtract operations.
b) Verify AND, OR, and EXOR operations.
c) Verify left and right shift operations.
d) Verify any other operations which exist.

The ALU in the 9900A is similar to a 74181 and performs the following operations on the A and B inputs.

0	A * B
A * $\overline{B}$	A plus $\overline{B}$ plus 1
$\overline{B}$	A
$\overline{A}$ * B	B
$\overline{A}$	A + B
A $\oplus$ B	All 1's
A plus B	

All of the operations it can perform are used at least twice during the test sequence. Many of the operations are used more often because of the extensive use of the ALU during the fetching and execution of the different op codes. Since a large amount of data passes through the ALU it is probable that all stuck at faults that could affect arithmetic operations are detectable and an in depth analysis was not done. However, the logical operations are not used as extensively so these were evaluated. It was found that all of the logic functions except A$\overline{B}$ were thoroughly checked for stuck at faults. Bit positions 1, 5, 6, 9, 10, 11, 13, 14 and 15 never had 00 applied and bit positions 9, 11, 13, and 15 never had 10 applied. Vendor F agreed to add tests to apply these conditions.

The four shift op codes (SRA, SRL, SRC, and SRL) use the DMUX, the SD register, and the shift counter which keeps track of the number of shifts. The four instructions were each used at least twice with both ones and zeroes shifted in a variety of patterns to verify the left and right shift operations. Since the SRA instruction fills the vacated positions with the original most significant bit (MSB) of the word, it was verified that this instruction was done with both a one and a zero in this position.

The shift count can be obtained from the C field of the instruction or bits 12 - 15 of workspace register zero (WR0) if C = 0. If bits 12 - 15 of WR0 equal 0 then the shift count is 16. The shift operations performed used all three methods of obtaining the shift count.

3. Conditional branch logic

a) Execute all jumps for both a true and a false condition. The flags not under test should be such that, if they are incorrectly selected, the jump is not performed.

The jump instructions result in a new value being loaded into the program counter if the bits of the status register are at the specified values. The conditions required for the various jumps to occur are shown in Table 3-1.

Analysis of the jump instructions revealed that two of them were not completely tested with respect to execution for both the true and false conditions. A jump high (JH), which requires that $ST_0 = 1$ and $ST_2 = 0$, was never attempted with $ST_0 = 1$ and $ST_2 = 1$ to verify that a jump would not occur. The jump low or equal (JLE) instruction, which requires that $ST_0 = 0$ or $ST_2 = 1$, was also not thoroughly checked. The jumps that were performed, occurred when $ST_0 = 0$ and $ST_2 = 0$ and when $ST_0 = 0$ and $ST_2 = 1$. This did not verify that just $ST_2 = 1$ was required to cause a jump.

Analysis also revealed that the dependence between the status register bits and jump instructions was not shown. A jump should or should not occur with a change in a bit position while all other status bits remain in the same state. This requirement was met by some but not all of the jump tests.

The last item analyzed in this area was the displacement associated with each jump operation. This displacement, which consists of bits 8 through 15 or the instruction, is transferred from the instruction register to the ALU, multiplied by two, and then added to the program counter. It was verified that during the execution of the jump instructions each of the bit positions in the displacement assumed both a one and a zero state to detect any stuck at faults. Independence of the data path between bits 8 through 15 or the instruction register and the ALU was checked by the op codes that used any of the registers or required a displacement or shift count.

Vendor F agreed to add the tests required to thoroughly check the execution of the jump instructions and to verify the relationship of the status register bits and the jump instructions.

MNEMONIC	OP CODE 0	1	2	3	4	5	6	7	MEANING	ST CONDITION TO LOAD PC
JEQ	0	0	0	1	0	0	1	1	Jump equal	ST2 = 1
JGT	0	0	0	1	0	1	0	1	Jump greater than	ST1 = 1
JH	0	0	0	1	1	0	1	1	Jump high	ST0 = 1 and ST2 = 0
JHE	0	0	0	1	0	1	0	0	Jump high or equal	ST0 = 1 or ST2 = 1
JL	0	0	0	1	1	0	1	0	Jump low	ST0 = 0 and ST2 = 0
JLE	0	0	0	1	0	0	1	0	Jump low or equal	ST0 = 0 or ST2 = 1
JLT	0	0	0	1	0	0	0	1	Jump less than	ST1 = 0 and ST2 = 0
JMP	0	0	0	1	0	0	0	0	Jump unconditional	unconditional
JNC	0	0	0	1	0	1	1	1	Jump no carry	ST3 = 0
JNE	0	0	0	1	0	1	1	0	Jump not equal	ST2 = 0
JNO	0	0	0	1	1	0	0	1	Jump no overflow	ST4 = 0
JOC	0	0	0	1	1	0	0	0	Jump on carry	ST3 = 1
JOP	0	0	0	1	1	1	0	0	Jump odd parity	ST5 = 1

TABLE 3-1 CONDITIONS REQUIRED FOR JUMP INSTRUCTIONS

4. Instruction register and decoding logic

   a) Execute each instruction type at least once.

   b) Use all addressing modes and all combinations of address modifiers.

   c) Access all used locations in the ROM.

   d) Use all of the constants in the constants block.

   e) Verify that each type of illegal op code is recognized as such by the device.

   f) Verify the integrity and independence of each bit.

The instruction set for the 9900 consists of 69 basic operations. When the different addressing modes and combinations of address modifiers are taken into account, a large number of variations are available. Evaluation of a functional test revealed that each of the basic operations was performed at least once, but not all of the variations were. However, it is not necessary to execute all of them to thoroughly test the 9900 because of the implementation of the logic.

The TS and TD fields of the instruction specify the addressing modes to be used and the S and D fields specify the address modifiers. Table 3-2 lists the addressing modes available. The instructions for the 9900 are decoded in groups; e.g., dual operand instructions are a group. To completely test the 9900, each of the five addressing modes should be tested for each group. This requirement was met.

Symbol	TS/TD	S/D	Mode
R	00	0-15	Workspace Register
*R	01	0-15	Workspace Register Indirect
@ Label	10	0	Symbolic
@ Table (R)	10	1-15	Indexed
*R +	11	0-15	Workspace Register Indirect Autoincrement

TABLE 3-2 9900A ADDRESSING MODES

All combinations of address modifiers were not used. However, it is not necessary to do so. During execution of an instruction the modifier is fetched from the appropriate field of the instruction register and added to the base address by the ALU to obtain the final required address. Since the ALU is used extensively throughout the execution of the instructions it is highly probable that all stuck-at-faults in it are detected.

It is necessary to have each bit assume a one and a zero state in each instruction field used for the modifiers to ensure that the connecting lines from the instruction register to the ALU are free of stuck-at-faults. In addition it is necessary to verify the independence of these lines. These requirements were met.

The ROM in the 9900 contains 256 addresses. Evaluation of the functional test revealed that approximately two-thirds of the addresses were accessed. Vendor F verified this and stated that the remaining addresses were not used. The only way they can possibly be accessed would be during a power up without reset. If this occurred, the program counter would be incremented by two and another instruction would be fetched during the next two clock cycles. There is no way to force the 9900 to access the unused addresses.

An analysis was performed by Vendor F to verify that if the ROM bits were stuck at the incorrect value they would be detected by the submitted test. From the analysis it was determined that additional instructions and asynchronous inputs are required. These tests were added.

Vendor F was asked for information concerning the contents and testing of the constants block. They stated that the constants are stored in a PLA and are routed from it to the B bus of the ALU. All of the constants were used a minimum of two times. The constants are used in additions by the ALU and any stuck-at-fault should be detected by incorrect addresses or data errors during memory storage.

The 9900 has a number of illegal op codes which are grouped by format type as are the legal ones. When an illegal op code is applied, the program counter should be incremented by two and the next instruction fetched. An illegal op code is included in the test for each decoded illegal format type to verify proper operation of the 9900.

A store status instruction had to be added after two of the illegal op codes to verify that the status register does not get changed by them. The integrity and independence of each bit in the instruction register were verified in the test.

5. Multiplexer verification

   a) Verify that each multiplexer can pass a one and a zero in each selection position with the other data inputs in the complement state.

      It was verified that all multiplexers are exercised in valid data transfers. It was not confirmed that a zero and a one appear on each data input line for each multiplexed selection. The probability that this happens, however, is very high due to the volume of data that is transferred within the 9900.

6. Status register

   a) Verify the integrity and independence of each bit.

   b) Verify that each bit can be set and reset by each op code that should affect it.

   c) Verify that certain op codes do not affect the status register.

The status register contains the interrupt mask level and information pertaining to the instruction operation. Evaluation of the test sequence verified that each status bit is set and cleared with the circuitry that can affect it with two exceptions. ST3 was never set to a one with the ABS instruction and ST12 is not reset to a zero for a RESET instruction. Although the parity bit (ST5) was verified after each op code that should affect it, the exclusive OR logic that generates it was not thoroughly tested. Generation of the parity bit with C8 as the input to the exclusive OR logic would complete the test of this circuitry.

It was learned from Vendor F that bits 7 through 11 which are defined as "not used" do not exist in the device. The independence of the remaining bits was verified in the test. However, it was found that the integrity of ST12 was not verified since it did not go through

the 1→1 transition. The store status (STST) instruction was performed after all of the instructions that were supposed to affect the status register bits. However, the instructions that are not supposed to affect the status register did not all have a STST instruction performed after them. Vendor F agreed to add tests which verify the effect of the ABS and RESET instructions on the status register, to verify the integrity of ST12 and to complete the test of the parity bit exclusive OR logic. They also agreed to add STST instructions after the remaining op codes that are not supposed to affect the status register. These include CKON, CKOF, CLR, LREX, STWP and two of the illegal ones.

7. Interrupt circuitry

   a) Verify that interrupt are acknowledged properly and ignored if they occur outside of an acceptable sample time.

   b) Verify the proper operation of the priority encoding circuitry for the interrupts.

   c) Verify that the addresses for the sixteen interrupt vectors are generated properly.

To fully test interrupts it is necessary to apply the interrupt request and execute the interrupt during the following op codes:

1) SBZ or SBO

2) TB

3) One of the jump op codes

4) B

5) LWPI

6) RTWP

7) LDCR

8) An illegal instruction going through ROM location 39 which can be reached by the execution of 0334 in hex.

In the submitted test, the interrupts were applied during STST, LIMI, BL, BLWP, IDLE, and XOP. Vendor F agreed to add one executed interrupt during an op code from each of the above sets. The 9900 should not acknowledge an interrupt if it occurs during the XOP or BLWP instructions or before the execution of the first instruction of an interrupt service routine. In the test, interrupts were attempted but not acknowledged for all three conditions.

The interrupt priority logic determines whether an interrupt will be acknowledged. Included in this logic are an interrupt level register and a difference arithmetic unit. When an interrupt is not occuring the interrupt level register contains the difference between the interrupt code input (IC_0 - IC_3) and status bits 12 - 15. If an interrupt is occuring, the interrupt level register contains the interrupt code input minus one which is eventually stored in the status register. During the test, status bits 12 - 15 are often first set to one less than the interrupt code level inputs to inhibit the interrupt. Then status bits 12 - 15 are changed to the interrupt code level to allow the interrupt to occur.

Three requirements have to be met to completely test the interrupt priority logic. First, all bit positions (ST12 - ST15 and IC_0 - IC_3) must have a value of one and zero for the recognition of an interrupt and the inhibition of an interrupt due to the mask level. This requirement is met by the test. The second requirement is that an interrupt is recognized when the status mask equals the input level and each bit position assumes a one and then a zero state. This requirement is also met. The third requirement is that an interrupt is recognized and an interrupt is inhibited for each of the following interrupt input level - mask level pairs:

Bit position	12 or 0	13 or 1	14 or 2	15 or 3
	S	S	S	D
	S	S	D	X
	S	D	X	X
	D	X	X	X

where: S means that the interrupt input bit is equal to the corresponding mask bit.

D means that the interrupt input bit is different from the corresponding mask bit.

X means a don't care bit correspondence.

All of these conditions are used in the test sequence except for the recognition of an interrupt with SSSD pair correspondence. Vendor F agreed to add a test for this condition. The interrupt vector address is generated by shifting the interrupt code input two places to the left for the workspace pointer address. The address is then passed through the ALU. Vendor F stated that the generation of all sixteen interrupt vector addresses is not required since the same circuitry is used for all addresses and is thoroughly tested.

8. Communications register units (CRU) Interface

a) Verify the integrity and independence of the bits in the shift register.

b) Verify the loading and counting of the counter which controls the number of bits shifted.

The CRU instructions LDCR and STCR are used to serially transfer data to or from a peripheral device. The starting address is contained in bits 3 through 14 of WR12. As the bits are transferred serially, the CRU address is incremented with each bit transfer. The contents of WR12 are not affected by the transfer. The C field of the instruction specifies the number of bits to be transferred. If eight or fewer bits are transferred the source address is a byte address. If nine or more bits are transferred, the source address is a word address. Shifts of less than eight bits and greater than eight bits, including shifts of 16 bits (C = 0), were performed. In addition, it was verified that the contents of WR12 were not altered by the shifts.

The MQ shift register is used for the serial transfer during the STCR and LDCR instructions and for shift operations during the MUL and DIV instructions. Its bit integrity and independence were checked thoroughly during these instructions. The shift counter is used to con-

trol the number of bits that are transferred. During the test it counted through all 16 counts and was loaded with count inputs such that each of the four bit positions was a one and a zero for at least one count. This completely tests it.

9. Register verification

    a) Verify the integrity and independence of the bits in all other registers - e.g., memory address, PC, WP, SD, DD, DA, and others that are transparent to the user.

    A thorough evaluation of the memory address, SD, DD, and DA registers was not performed with respect to bit independence and integrity. These registers change state many times during the test since the operands are obtained from many memory locations and pass through the SD, DD, and DA registers before or after they are operated on by the processor. This high volume of data movement results in a high probability that the independence and integrity of these registers is verified.

    Since the contents of the WP and PC did not change as often during the test, these registers were thoroughly evaluated. The workspace pointer required additional tests to completely verify the independence of the seven most significant bits (0-6) and the integrity of bits 1, 3, 5, and 14. The program counter required additional tests to completely verify the independence of the six most significant bits (0-5) and the integrity of bits 1, 3, 5, and 6. Vendor F agreed to add these tests.

    The 9900 contains no registers that are transparent to the user.

10. Control functions

    a) Verify the operation of all external control inputs.

The control inputs that have not been previously discussed are $\overline{\text{HOLD}}$, $\overline{\text{LOAD}}$, and READY. $\overline{\text{HOLD}}$ is checked in several places under varying conditions, to thoroughly verify its operation. This includes the activation of $\overline{\text{HOLD}}$ before or during a CRU instruction which checks the timing relationship between HOLD and CRU addressing. If $\overline{\text{HOLD}}$ becomes active during a CRU operation, an extra clock cycle is required after its deactivation to reassert the CRU address before the

serial bit transfer can occur. $\overline{\text{LOAD}}$ was also checked under varying conditions to verify its operation. These include activating $\overline{\text{LOAD}}$ when RESET is active to verify that the $\overline{\text{LOAD}}$ trap will occur after the reset function and activating $\overline{\text{LOAD}}$ to end an idle state. The idle state can also be terminated by an interrupt or $\overline{\text{RESET}}$. Both of these conditions are tested. The READY input was also checked under varying conditions to verify its operation.

11. Data bus verification

    a) Verify that only one block at a time is driving the internal bus by placing complemented data in the blocks not driving the bus.

    b) Verify the independence of each data line.

The internal data bus is used extensively throughout the test. No analysis was done to verify that each line was in a state opposite to the others or that the blocks not driving it were in a state opposite to the driving one. Due to the excessive volume of data that passes over the lines the probability is high that the independence of data lines and data source blocks is shown.

12. Miscellaneous verifications

    a) Verify independence of any data path not done previously.

    b) Verify IC pin independence; adjacent pins as a minimum.

    c) Verify high impedance state of the memory address and data buses.

    d) Verify the proper generation of all sixteen XOP trap vector addresses.

    e) Execute tests at maximum frequency to verify proper timing operation within the microprocessor.

Independence of all of the IC pins to each other was not analyzed. Since the data bus and memory address bus are used extensively there is a high probability that their independence was verified. Adjacent pin

independence was verified for the remaining pins.

The high impedance state of the memory address and data buses was verified during the $\overline{\text{HOLD}}$ tests.

The address for the XOP instruction is obtained by obtaining the D field from the instruction register, shifting it left two places, and then adding the result to 40_{16}. Since several other instructions use the D field the logic involved is completely tested without generating all 16 XOP trap vector addresses.

Vendor F performs the functional test at 1 MHz to detect stuck-at-faults. The functional test patterns are also used for the AC tests which are done with worst case timing for the device.

Summary of Added Tests

The following is a summary of the tests that were added to the functional test.

1. Tests to check the control outputs on all of the vectors.

2. Tests to check the JH and JLE instructions completely.

3. An ABS instruction that sets ST3 to a one and a RSET instruction that resets ST12 to zero.

4. STST instructions after two of the illegal op codes.

5. A test to check the SSSD pair correspondence for the interrupt logic.

6. Jump tests to check the status register-jump integrity relationship.

7. Tests to completely check the SZC and SZCB instructions. These op codes use the ALU's A OR B and A AND B functions.

8. A parity of a byte (C8) to completely check the exclusive OR logic that generates the parity bit.

9. An executable interrupt during each of the following sets of instructions.

a) SBZ or SBO

b) TB

c) One of the jump op codes

d) B

e) LWPI

f) RTWP

g) LDCR

h) An illegal instruction going through ROM location 39 which can be reached by execution of 0334 in hex.

10. Instructions to perform the following so that stuck-at ROM bits can be detected.

a) An instruction that uses indexed addressing in which the indexed register contents are negative, i.e., @ Table (R) where (contents of R) < 0.

b) An illegal op code ODEF, 0123, or 079A located at a memory location greater than 2000 in hex.

c) An executed jump to and from a memory location greater than 2000 in hex.

d) An X instruction for which the WP points to a memory location greater than 2000 in hex.

e) An AI instruction located in a memory location greater than 2000 in hex and with negative data for the immediate value.

f) An interrupt that executes after an instruction located in a memory location greater than 2000 in hex, with the WP trap vector pointing to an address greater than 2000 in hex and with ST (0, 1 or 2) = 1 prior to the interrupt.

g) A LOAD with ST (0, 1 or 2) = 1.

h) A DIV for which the quotient's last bit is a 0 and the remainder is greater than 2000 in hex.

i) An illegal op code 0334 located at a memory location greater than 2000 in hex.

11. STST op codes after CKON, CKOF, CLR, LREX, and STWP instructions.

12. Tests that have all ST bits equal to a one prior to a hardware reset.

13. Changes to check independence and integrity of the WP and PC.

14. An XOP requiring an XOP to check ST12 integrity.

15. An X instruction followed by an X instruction.

SECTION IV

TEST DEVELOPMENT FOR THE 6821 PERIPHERAL INTERFACE ADAPTER

Objective

An evaluation was made of the proposed tests which Vendor G submitted for use in the slash sheet for the 6821. The approach used to analyze these tests was the Procedure for LSI Test Development which is documented in RADC report RADC-TR-78-138. The block diagram used for the functional test evaluation is from Vendor G's catalog and is shown in Figure 4-1.

Evaluation of the Functional Test

Summary

Vendor G performs a dynamic functional test consisting of three computer generated patterns (Patterns 1, 2, and 3) each containing 570 vectors. Pattern 1 is used to perform all of the functional and most of the switching speed tests for the device. Patterns 2 and 3 are performed with the same input conditions as Pattern 1 but the outputs are strobed at different times and levels to verify the device's CMOS drive capability.

A list of tests required to check the 6821 was developed. It was used, along with the Procedure for LSI Test Development, as the basis of the evaluation. Most of the requirements of these two checklists were met. However, the following discrepancies existed:

1. Not all of the stuck-at-faults for the Chip Select and Read/Write Control block could be caught.

2. Register independence was not completely verified.

3. Register bit independence was not completely tested.

4. The high-impedance capability of the PB Bus was not verified.

GE developed vectors to correct these deficiencies. These vectors were appended to each of the three patterns.

Discussion

The 6821 is used to interface the 6800 microprocessor to peripherals through two 8-bit bidirectional peripheral data buses, Peripheral A (PA)

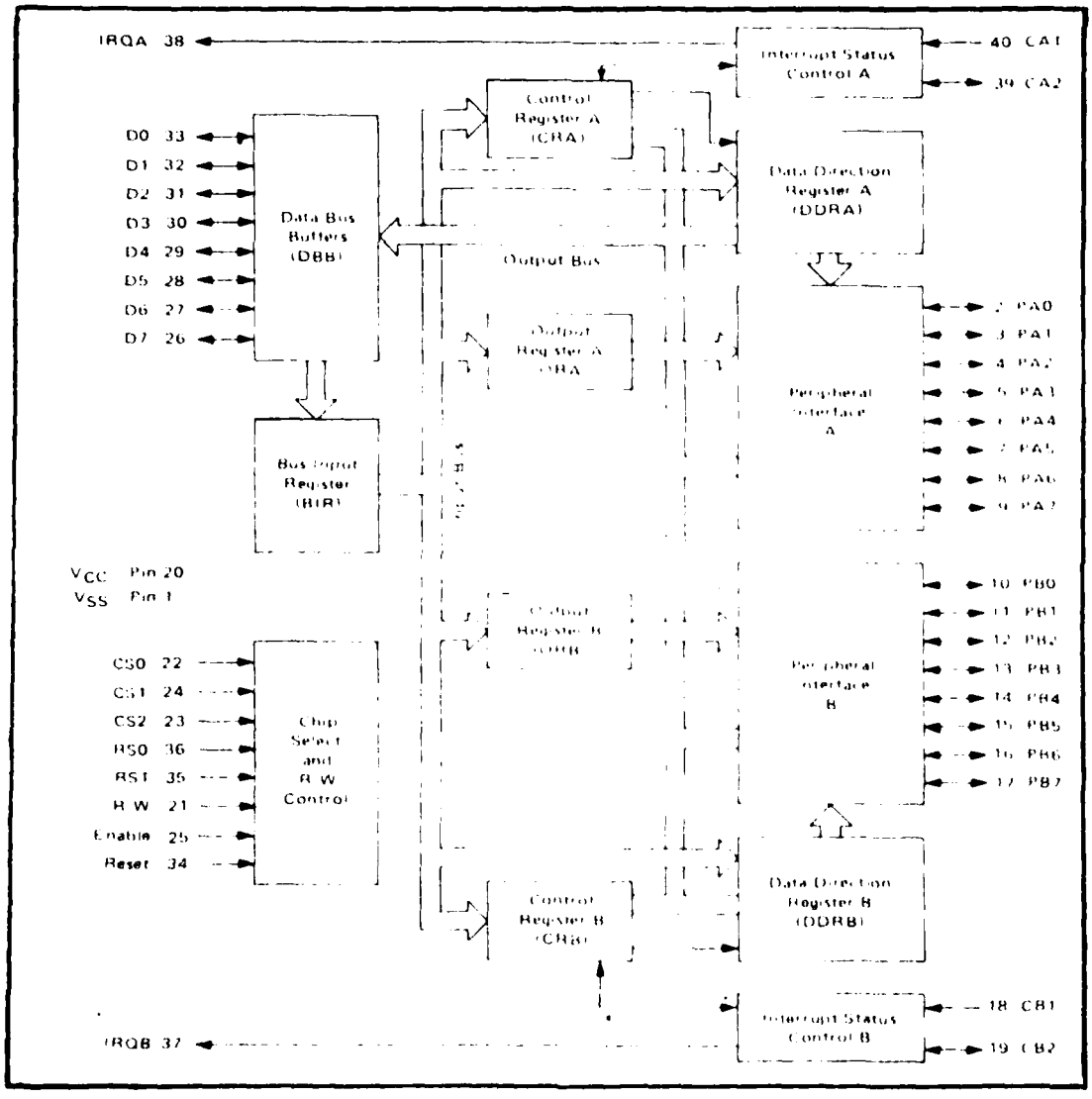


FIGURE 4-1 BLOCK DIAGRAM OF THE 6821

Bus and Peripheral B (PB) Bus, and four interrupt/control lines. Each peripheral line can be programmed to act as an input or output and each of the four interrupt/control lines (CA1, CA2, CB1, CB2) can be programmed to operate in an interrupt mode, a handshake mode, a program-controlled set/reset output mode, or a pulse strobed mode. The interface to the CPU is through an 8-bit bidirectional Data Bus (DB).

The 6821 contains seven registers as shown in Figure 4-1. They are the Bus Input Register (BIR), Control Register A (CRA), Control Register B (CRB), Data Direction Register A (DDRA), Data Direction Register B (DDRB), Output Register A (ORA) and Output Register B (ORB). During a write operation, data is transferred from the Data Bus lines to the BIR on the rising edge of the ENABLE pulse. On the trailing edge, the data is transferred from the BIR to one of the other six registers, the selection of which depends on the two register select lines (RS0 and RS1) and the state of bit position 2 in the control registers. This results in a master-slave operation where the BIR is the master and any of the other six registers is the slave.

During a read operation, the peripheral data appears directly on the DB lines when the device is selected. The data paths from the Data Bus to the PA and PB Buses are similar except for the output buffers. The PA Bus is bidirectional but cannot actually enter a high-impedance (Hi - Z) state. Each of the lines on the PA Bus is configured as shown in Figure 4-2. In

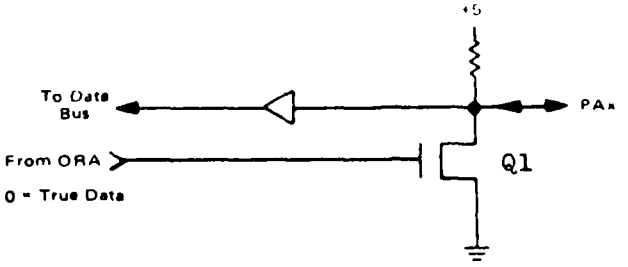


FIGURE 4-2 PERIPHERAL A BUS LINE

the input mode, Q1 is turned off and the line is pulled high by the internal pullup resistor. Data can now be transferred from the PA Bus to the Data Bus. The PA Bus is TTL and CMOS compatible. The PB Bus can enter a Hi - Z state and is only TTL compatible.

The tests required for a complete functional test of the 6821 must do the following:

1. Verify that the six slave registers are reset to zero when $\overline{\text{RESET}}$ is

applied.

2. Verify register independence for the six slave registers.

3. Verify bit independence and integrity for all seven registers.

4. Verify proper operation of the interrupt status control logic.

5. Verify proper operation of the Chip Select circuitry.

6. Verify proper operation of the Read/Write Control circuitry.

7. Verify proper operation of the PA and PB lines.

8. Verify data path independence.

The functional blocks and interconnecting data paths shown in Figure 4-1 were analyzed with respect to these requirements. The following describes the results of the evaluation:

1. Verify that the six slave registers are reset to zero when $\overline{\text{RESET}}$ is applied.

   To verify the reset function it is necessary that each register contain ones prior to the application of $\overline{\text{RESET}}$. The 6821 is reset eight times during the functional test. Each bit position in the registers is a one prior to at least one of the resets thus verifying proper operation.

2. Verify register independence for the six slave registers.

   The six registers and register select logic in the 6821 can be represented as shown in Figure 4-3. To verify register independence it is necessary that each bit position of each register assume a one and a zero state while all of the corresponding bits in the other registers either individually or collectively assume the opposite state. This verifies that the registers are unique and that writing into any one of them will not alter the contents of the others.

   The submitted test did not meet these requirements. The contents of the registers were verified without knowing if the others were actually in the complemented state and it was not verified that the registers were unique. The vectors added by GE to detect stuck-at-faults on the select gates for the six registers write a different pattern into each of the registers and then perform six successive reads to verify that they were written into correctly. The following sequence of tests were added to detect stuck-at-one faults on the transmission gate enable inputs:

   a) Write a pattern into a register and read it out.

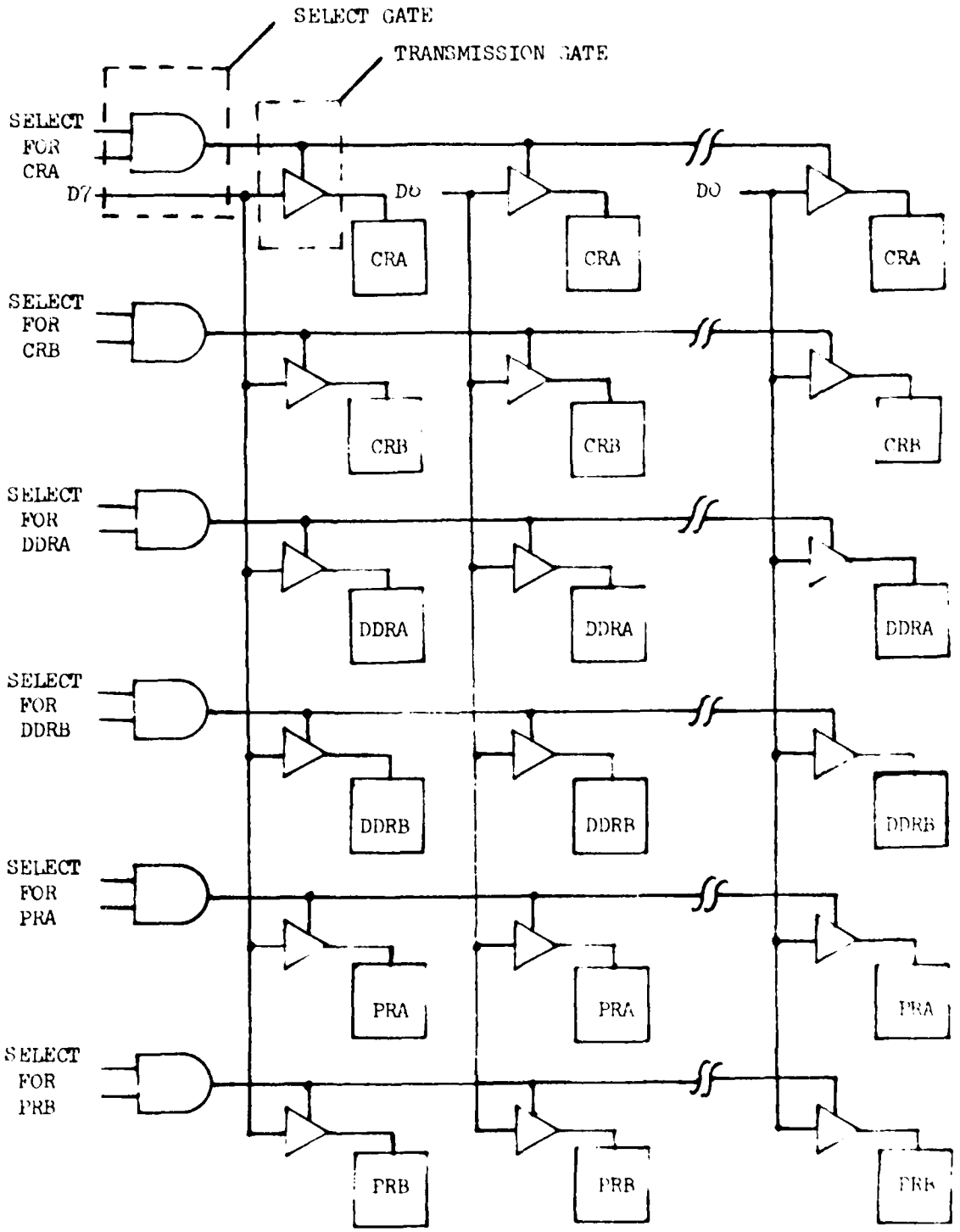


FIGURE 4-3 SLAVE REGISTERS AND SELECT LOGIC

b) Write the complement into any other register.

c) Read the contents of both registers to verify that the first was not changed and that the second contains the complemented pattern.

d) Repeat this procedure using each of the other registers as the starting register.

3. Verify bit independence and integrity for all seven registers.

Bit independence for CRA and CRB was not thoroughly tested. The charts in Figure 4-4 show which bit positions were checked. A blank square indicates that no independence check was made between the bit positions specified by the row and column. GE added the vectors necessary to completely test for bit independence.

When checking for cell integrity it was found that not all bit positions of the registers went through the 1→1 and 0→0 transitions. However, when the implementation of the latches was learned, it was determined that these transitions were not necessary.

4. Verify proper operation of interrupt status control logic.

The interrupt status control blocks include the logic for the interrupt input lines CA1 and CB1 and the bidirectional peripheral control lines CA2 and CB2. The bidirectional lines can operate as interrupt input lines or peripheral control lines. Tables 4-1 through 4-4 describe the operation of the four interrupt/control lines.

Pattern 1 completely verified the operation of CA1, CB1, CA2, and CB2 as interrupt inputs and of CA2 and CB2 as output control lines (driving TTL) for all of the modes shown in the tables. The conditions for IRQA and IRQB which are marked by an * in each of these tables were not included in the test. However, when Vendor G was contacted they stated that tests for these conditions were unnecessary because the logic involved is checked by the condition marked with the ** in the same table.

5. Verify proper operation of the Read/Write Control circuitry.

The ENABLE signal provides the timing to the PIA. During a write operation, data is transferred from the Data Bus to the BIR on the rising edge of ENABLE and into one of the slave registers on the trailing edge. The ENABLE signal also provides the timing for a read operation by enabling the output tri-state buffers so that data from any of the slave registers or the peripheral lines will appear on the Data Bus.

During the functional test a pulse appears on the ENABLE input for every test vector. The outputs are compared after the rising edge of

Note: An X indicates that bit independence has been checked for the two corresponding bits for one of the two required conditions.

CRA

Bit in "0" State \ Bit in "1" State	0	1	2	3	4	5	6	7
0			X	X	X		X	
1	X		X	X	X	X	X	X
2							X	
3	X	X	X			X	X	X
4	X	X	X	X		X	X	X
5	X	X	X	X	X		X	X
6	X	X	X	X	X	X		X
7	X	X	X	X	X	X	X	

CRB

Bit in "0" State \ Bit in "1" State	0	1	2	3	4	5	6	7
0			X	X	X		X	
1	X		X	X	X	X	X	X
2				X			X	
3	X	X	X			X	X	X
4	X	X	X	X		X	X	X
5	X	X	X	X	X		X	X
6	X	X	X	X	X	X		X
7	X	X	X	X	X	X	X	

FIGURE 4-4 RESULTS OF BIT INDEPENDENCE EVALUATION

CRA-1 (CRB-1)	CRA-0 (CRB-0)	Interrupt Input CA1 (CB1)	Interrupt Flag CRA-7 (CRB-7)	MPU Interrupt Request $\overline{IRQA}$ ($\overline{IRQB}$)
0	0	↓ Active	Set high on ↓ of CA1 (CB1)	Disabled — $\overline{IRQ}$ remains high **
0	1	↓ Active	Set high on ↓ of CA1 (CB1)	Goes low when the interrupt flag bit CRA-7 (CRB-7) goes high
1	0	↑ Active	Set high on ↑ of CA1 (CB1)	Disabled — $\overline{IRQ}$ remains high *
1	1	↑ Active	Set high on ↑ of CA1 (CB1)	Goes low when the interrupt flag bit CRA-7 (CRB-7) goes high

Notes 1 ↑ indicates positive transition (low to high)

2 ↓ indicates negative transition (high to low)

3 The interrupt flag bit CRA-7 is cleared by an MPU Read of the A Data Register, and CRB-7 is cleared by an MPU Read of the B Data Register

4 If CRA-0 (CRB-0) is low when an interrupt occurs (interrupt disabled) and is later brought high, $\overline{IRQA}$ ($\overline{IRQB}$) occurs after CRA-0 (CRB-0) is written to a "one".

TABLE 4-1 CONTROL OF INTERRUPT INPUTS CA1 AND CB1

CRA-5 (CRB-5)	CRA-4 (CRB-4)	CRA-3 (CRB-3)	Interrupt Input CA2 (CB2)	Interrupt Flag CRA-6 (CRB-6)	MPU Interrupt Request $\overline{IRQA}$ ($\overline{IRQB}$)
0	0	0	↓ Active	Set high on ↓ of CA2 (CB2)	Disabled — $\overline{IRQ}$ remains high **
0	0	1	↓ Active	Set high on ↓ of CA2 (CB2)	Goes low when the interrupt flag bit CRA-6 (CRB-6) goes high
0	1	0	↑ Active	Set high on ↑ of CA2 (CB2)	Disabled — $\overline{IRQ}$ remains high *
0	1	1	↑ Active	Set high on ↑ of CA2 (CB2)	Goes low when the interrupt flag bit CRA-6 (CRB-6) goes high

Notes: 1. ↑ indicates positive transition (low to high)

2. ↓ indicates negative transition (high to low)

3. The interrupt flag bit CRA-6 is cleared by an MPU Read of the A Data Register and CRB-6 is cleared by an MPU Read of the B Data Register

4. If CRA-3 (CRB-3) is low when an interrupt occurs (interrupt disabled) and is later brought high, $\overline{IRQA}$ ($\overline{IRQB}$) occurs after CRA-3 (CRB-3) is written to a "one".

TABLE 4-2 CONTROL OF CA2 AND CB2 AS INTERRUPT INPUTS CRA-5 (CRB-5) IS LOW

CRA-5	CRA-4	CRA-3	CA2	
			Cleared	Set
1	0	0	Low on negative transition of E after an MPU Read "A" Data operation.	High when the interrupt flag bit CRA 7 is set by an active transition of the CA1 signal.
1	0	1	Low on negative transition of E after an MPU Read "A" Data operation.	High on the negative edge of the first "E" pulse which occurs during a deselect.
1	1	0	Low when CRA 3 goes low as a result of an MPU Write to Control Register "A"	Always low as long as CRA 3 is low. Will go high on an MPU Write to Control Register "A" * that changes CRA 3 to "one"
1	1	1	Always high as long as CRA 3 is high. Will be cleared on an MPU Write to Control Register "A" that clears CRA-3 to a "zero".	High when CRA-3 goes high as a result of an MPU Write to Control Register "A". ↓ **

TABLE 4-3 CONTROL OF CA2 AS AN OUTPUT
CRA-5 IS HIGH

CRB-5	CRB-4	CRB-3	CB2	
			Cleared	Set
1	0	0	Low on the positive transition of the first E pulse following an MPU Write "B" Data Register operation	High when the interrupt flag bit CRB-7 is set by an active transition of the CB1 signal
1	0	1	Low on the positive transition of the first E pulse after an MPU Write "B" Data Register operation.	High on the positive edge of the first "E" pulse following an "E" pulse which occurred while the part was deselected
1	1	0	Low when CRB-3 goes low as a result of an MPU Write in Control Register "B"	Always low as long as CRB-3 is low. Will go high on an MPU Write in Control Register "B" that changes CRB-3 to "one" *
1	1	1	Always high as long as CRB-3 is high. Will be cleared when an MPU Write Control Register "B" results in clearing CRB-3 to zero	High when CRB-3 goes high as a result of an MPU Write into Control Register "B". **

TABLE 4-4 CONTROL OF CB2 AS AN OUTPUT
CRB-5 IS HIGH

ENABLE. This prevents the detection of all stuck-at-one faults on gates to which the ENABLE signal is applied.

GE added a set of vectors which will verify that data is not transferred during a write operation if ENABLE is held low. This verifies that all stuck-at-one faults are detected on those gates to which ENABLE is applied during read and write operations.

The following sequence of vectors was added:

a) Write all 1's or 0's into a register and verify its contents using the PA or PB Bus or IRQA(B) as required.

b) Select another register and write the complement of a.

c) Hold ENABLE low on the next cycle and select the register in a. Using the PA or PB Bus or IRQA(B) as required, verify that the contents of the register in a have not changed.

   If the ENABLE input were stuck-at-one, the register in step a will be written, on step c, with the complemented data which is still in the Bus Input Register.

To check for a stuck-at-one during a read operation, the following test sequence was added:

a) Write all zeros into a register.

b) Read the register in a.

c) With ENABLE low verify that the data lines are in the Hi - Z state. If the ENABLE input were stuck-at-one, a zero would appear on the Data Bus.

6. Verify proper operation of the Chip Select circuitry.

The Data Bus in the 6821 will enter the Hi - Z state when the device is in the write mode, is deselected because either CS0, CS1, or $\overline{CS2}$ is inactive, or is in the read mode and ENABLE is low. The submitted test did not verify that the Data Bus is in the Hi - Z state for all of these conditions resulting in undetected faults in the Chip Select and Read/Write Control logic. GE added the vectors required to completely test this circuitry.

7. Verify proper operation of PA and PB Buses.

Each of the lines on the PA and PB Buses can be programmed to act as an input or an output by the corresponding bits in the Data Direction registers. In the Input mode, data is transferred from the Peripheral Bus to the Data Bus. In the Output mode, data is transferred from the Data Bus to the Peripheral Bus via the Output Registers. The contents

of the Output Registers can also be read from the Data Buses by the microprocessor. The test verified the operation of the PA and PB Buses. It was shown that each line on the PA and PB Buses could be programmed as an input and transfer both a one and a zero to the Data Bus. In addition, both a one and a zero were read from all of the Output Register bits when the PA and PB lines were programmed as outputs.

8. Verify data path independence.

Independence of the data paths from the various registers to the Data Bus, PA Bus, and PB Bus was verified.

Evaluation of the Switching Speed Tests

Vendor G performs a dynamic functional test on the 6821. In this way the switching speed parameters are verified at the same time that the functional test is done and all gates can contribute to the delays. Most of the propagation delays and all of the setup and hold times are verified using Pattern 1. Patterns 2 and 3 are used to check the propagation delay times when the PA Bus and CA2 interface with CMOS. Pattern 4 is run at 1MHZ to verify maximum frequency operation and is then repeated at 1KHZ to show that the device is static.

SECTION V

TEST DEVELOPMENT FOR THE 2909 MICROPROGRAM SEQUENCER

Objective

The purpose of this evaluation was to review the tests which Vendor A had submitted for inclusion in the slash sheet for the 2909. The approach used was the Procedure for LSI Test Development which is documented in RADC Report RADC-TR-78-138 and will not be repeated here. The block diagram for the 2909 is shown in Figure 5-1.

Summary of the Functional Test Evaluation

Vendor A provided GE with a gate-level logic diagram, listings of their functional and switching speed test patterns, and a copy of their test program. A review of their functional test, which contained 349 vectors, revealed that it met most of the requirements of the checklist in the Procedure for LSI Test Development but was deficient in the following three areas:

1. IC pin and data path independence were not thoroughly checked. The OR inputs were either all zeroes or all ones. The even and odd bits of the D inputs were always in the same state. The independence of $\overline{OE}$ to the other pins was only partially checked because it was held at a logic zero during the entire test. Consequently, verification of the high-impedance capability of the Y outputs was not done.

2. Nineteen stuck-at-faults in the multiplexer and S1, S0 circuitry and two in the instruction register circuitry were not detected.

3. Bit independence of the even and odd bits of each word in the stack was not thoroughly checked.

During the course of the review, Vendor A indicated that they had added all of the switching speed test patterns to their functional test and requested that these tests also be added to the functional test in the slash sheet. This increased the number of test vectors to 826.

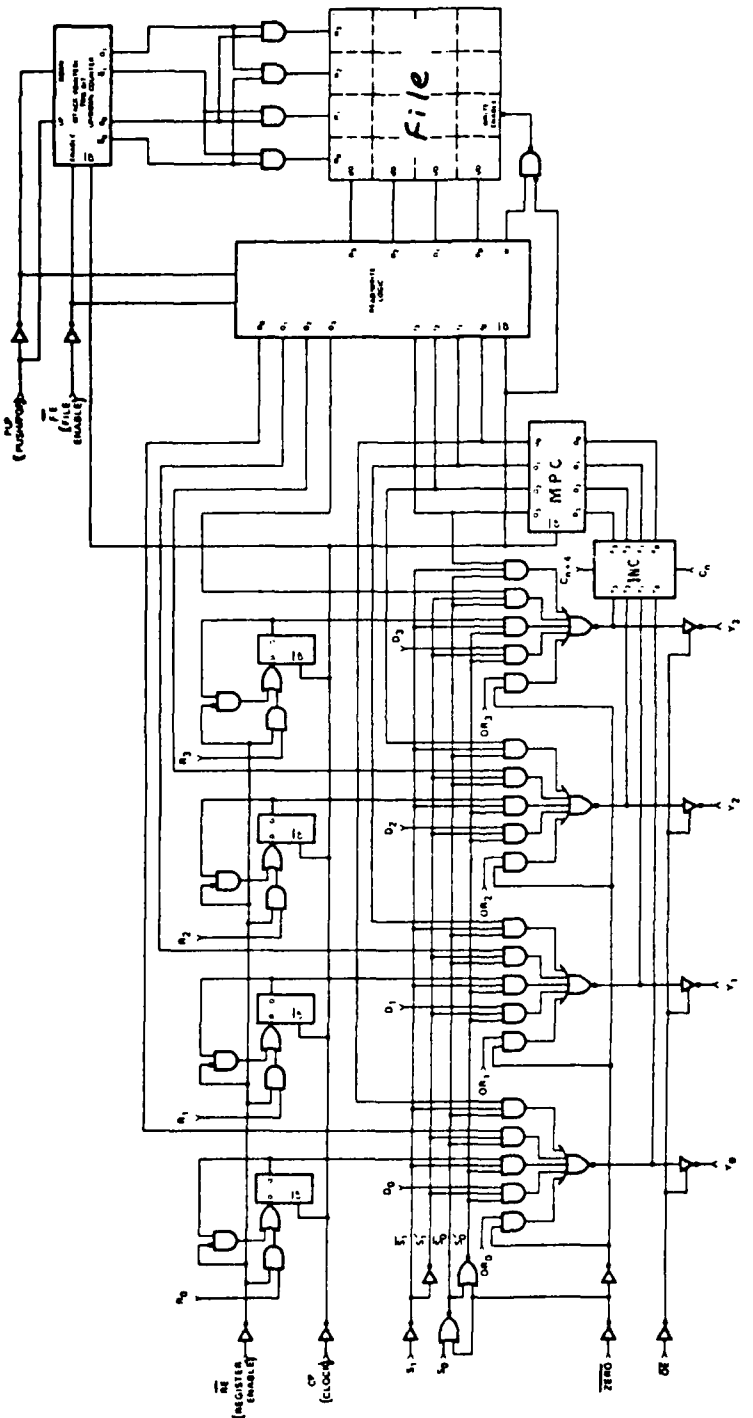


Figure 5-1 2909 Block Diagram

The addition of these patterns corrected some of the deficiencies and additional vectors developed by GE corrected the remaining ones.

Discussion

Vendor A partitioned the 2909 into the following three functional blocks: the Multiplexer and Instruction Register, the Incrementer and Program Counter, and the Push-Down Stack. Copies of the tests for each block and a magnetic tape containing the entire Sentry test program were received from Vendor A. The vectors for the functional test were extracted from the program and reformatted on GE's Tektronix tester. This allowed the pin data to be arranged in a more desirable order and facilitated evaluation of the functional test.

The Procedure for LSI Test Development was used as a guideline for determining the test requirements. Since a gate-level logic diagram was also received from Vendor A, the test evaluation included a review of the test vectors to determine stuck-at-fault coverage. A brief description of the tests for each functional block and the results of GE's evaluation are given below.

1. Multiplexer and Instruction Register

   The multiplexer was tested by passing 0101 and 1010 through each of the four normal inputs: Instruction Register, Stack, Program Counter, and Direct. The priority of the OR inputs over the four normal inputs and the priority of the $\overline{\text{Zero}}$ inputs over the OR and four normal inputs were verified. A "1" and a "0" were passed in each selection position, but the Y outputs were not always sensitized to detect a failure and 19 stuck-at-faults were not detected. A block diagram that shows these faults is given in Figure 5-2. A list of the modified and additional vectors required to detect them is given in Table 5-1.

   The instruction register was tested by first loading in 0101 with $\overline{\text{RE}}$ low. Then 1010 was applied to the inputs of the register with $\overline{\text{RE}}$ high, the register was clocked, and the contents were checked to ensure that 0101 remained in the

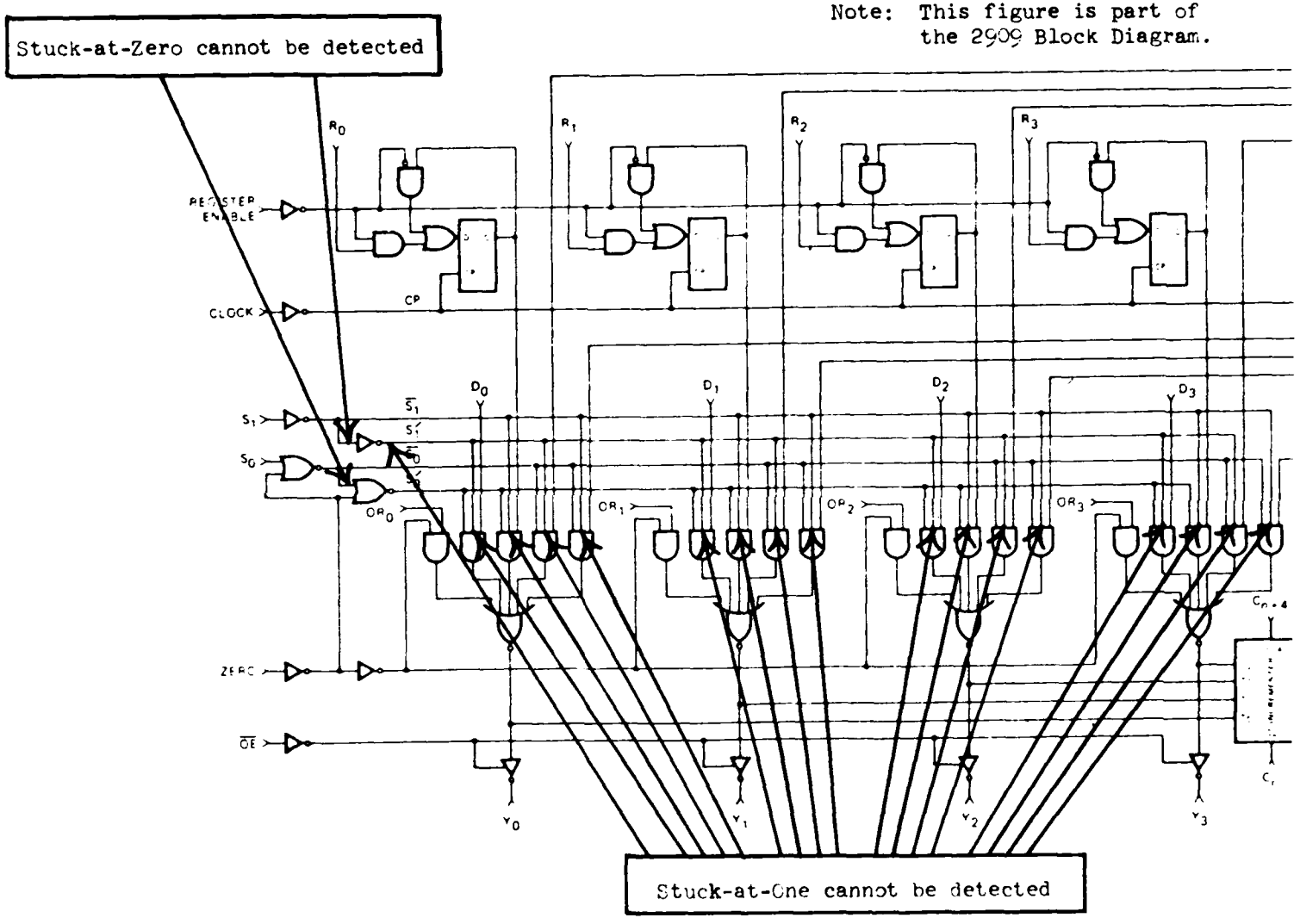


Figure 5-2 Undetected Multiplexer Stuck-at-Faults

Vector Numbers	Modifications
[illegible]	Change R from 0000 to 0101 and CP from 0 to 1
3	Change R from 0000 to 1010 and CP from 0 to 1
[illegible]	Change D from 0000 to 1010
1[illegible]	Change D from 0000 to 1010
22	Change D from 0000 to 0101
33	Change D from 0000 to 0101

Additions

Pin Number	1	2	3	4	5	6	[illegible]	[illegible]	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
22A	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0	0
23A	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	0	0	0	1	0	0
30A	1	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1	1	1	0	1	0	0	0	0	0	1	1	0
31A	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	1	0	0
32A	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	0	1	0	0

Table 5-1 Modified and Additional Vectors to Detect Stuck-at-Faults in the Multiplexer

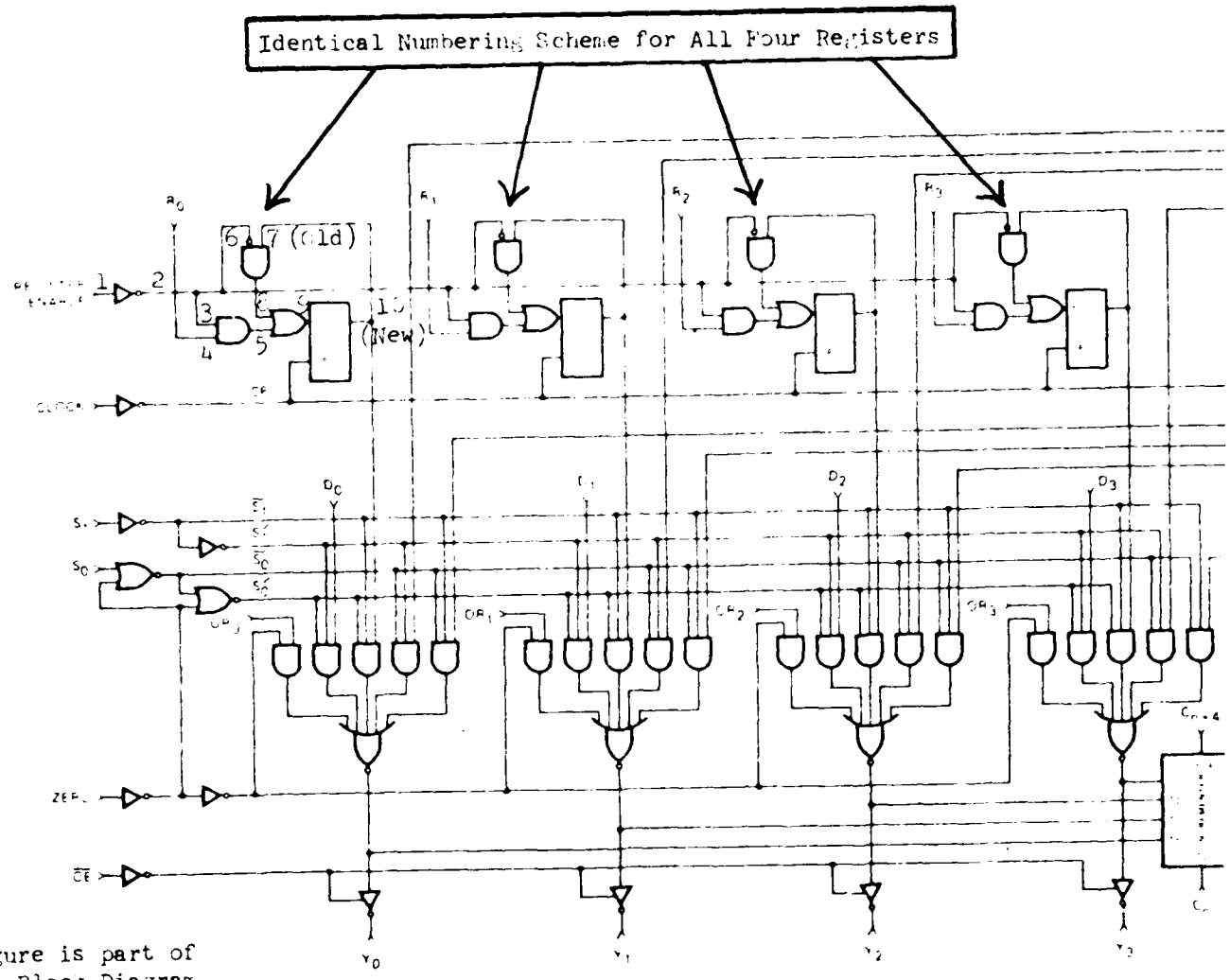
register. $\overline{RE}$ was made low and 1010 was loaded into the register. These tests were mixed with the multiplexer tests. The instruction register contains three gates and one D flip-flop for each bit and input buffers on the $\overline{RE}$ and clock inputs. The circuit node designations are shown in Figure 5-3. The eight possible states that the circuitry for each bit can assume are given in Table 5-2. It was found that stuck-at-zero faults were not detected on one input of an AND gate for the R_1 and R_3 bits. One vector was modified to detect these faults. Table 5-3 shows the results of the evaluation.

2. Incrementer and Program Counter

The incrementer was tested by having it count from 0 to 15 and then to 0 on the next count by keeping the carry-in input high. This resulted in a "1" and a "0" being passed through each input of the program counter. There are 64 combinations that can be applied to the incrementer and program counter from the six inputs: the clock, the carry-in, and the four multiplexer inputs. All 64 combinations are applied and the Y outputs are sensitized to detect any stuck-at-failures. The purpose of test vectors 514 through 534, which again forced the incrementer to count from 0 to 15, was not apparent. Vendor A was questioned but could not explain why the tests were repeated. The tests were not deleted in the event they checked a device sensitivity.

$\overline{RE}$	R	Old Q (.)	Condition Designation
0	0	0	A-0
0	0	1	A-1
0	1	0	B-0
0	1	1	B-1
1	0	0	C-0
1	0	1	C-1
1	1	0	D-0
1	1	1	D-1

Table 5-2 The Eight States of the Instruction Register Circuitry



Note: This figure is part of the 2909 Block Diagram.

Figure 5-3 Circuit Node Designations for the Instruction Registers

Vector # (CP = 1)	Condition	Fault	Node Number	1		2		3		4		5		6		7		8		9		10		Outputs Sensitized
				S-A-0	S-A-1	S-A-0	S-A-1	S-A-0	S-A-1	S-A-0	S-A-1	S-A-0	S-A-1	S-A-0	S-A-1	S-A-0	S-A-1	S-A-0	S-A-1	S-A-0	S-A-1	S-A-0	S-A-1	
												R_3 and R_1												
4	*A - ?									x		x						x		x		x		No
6	A - 0									⊗		⊗						⊗		⊗		⊗		Yes
13	D - 0			⊗		⊗		⊗				⊗				⊗		⊗		⊗		⊗		Yes
17	D - 0			x		x		x				x				x		x		x		x		No
22	B - 0				⊗	⊗			⊗	⊗			⊗								⊗		⊗	Yes
27	C - 1			⊗			⊗								⊗	⊗		⊗		⊗		⊗		Yes
33	C - 1			x		x								x		x		x		x		x		No
												R_2 and R_0												
4	*A - ?									x		x						x		x		x		No
6	B - 0				⊗	⊗		⊗		⊗		⊗								⊗		⊗		Yes
13	C - 1			⊗			⊗								⊗	⊗		⊗		⊗		⊗		Yes
17	C - 1			x		x								x		x		x		x		x		No
22	A - 1				⊗	⊗				⊗		⊗		⊗					⊗		⊗		⊗	Yes
27	D - 0			⊗			⊗		⊗			⊗					⊗	⊗		⊗		⊗		Yes
33	D - 0			x		x		x				x				x		x		x		x		No

* The previous outputs of the registers are not known. Therefore, only faults which are not dependent on the previous output of the D flip-flop (Old Q -- Node 7) are listed.

x - Outputs are not sensitized to pick up these faults.
⊗ - Outputs are sensitized to pick up these faults.

Note: The only faults that were not detected are S-A-0's on Node 6 for the R_3 and R_1 bits. The R input on Vector 4 was changed to 1010 to correct this deficiency. This forces condition A-1 for Vector 6 and will detect S-A-0's on Node 6 for R_3 and R_1.

Table 5-3 List of Stuck-at-Faults That Are Detected in the Instruction Register by the Functional Test

3. Push-Down Stack

The manufacturer's test for this functional block was started by pushing zeroes into each word of the stack. The zeroes were then popped out to ensure that the stack was initialized. Ones were then loaded into one word location with zeroes remaining in the other three.

All four words were popped, then another pop was done to show that the stack is a wrap-around type and to set the stack pointer to a new starting point for subsequent tests. The test was repeated three more times with ones loaded into a different word location each time and zeroes remaining in the other three. The contents of each word location during these tests are given in Table 5-4. The same four tests were then repeated with zeroes in one word location and ones in the other three. This sequence fully checked word independence in the stack.

Then a 0101 pattern was pushed into and popped out of one location at a time. This was repeated for each of the four word locations. Table 5-5 shows the sequence for these pushes and pops. Vendor A's tests did not thoroughly verify bit independence within each word. Vectors were added to repeat the above tests with 1001, 0110, and 1010 patterns which completes the tests for bit independence within each word. All combinations were applied to $\overline{FE}$ and PUP to thoroughly check the operation of the stack pointer.

4. General Tests and Comments

a) During the review of the functional test, Vendor A decided to add all of the switching speed test patterns to their functional test, thus increasing the functional test from 349 to 826 vectors. This decision was based on finding a device (tested by them) that exhibited a stuck-at-fault in the stack that was detectable by the switching speed test patterns but not the functional test patterns.

Operation	Stk	w_0	Stk	w_1	Stk	w_2	Stk	w_3	Vector	Vector*
Push	0	1111	1	0000	2	0000	3	0000	1e3	335
Pop	3	↓	0	↓	1	↓	2	↓	1e	341
Pop	2	↓	3	↓	0	↓	1	↓	1 1	343
Pop	1	↓	2	↓	3	↓	0	↓	1 3	345
Pop	0	↓	1	↓	2	↓	3	↓	1 5	34
Pop	3	1111	0	↓	1	↓	2	↓	1	351
Push	0	0000	1	0000	2	0000	3	0000	203	35
Push	1	0000	2	0000	3	0000	0	1111	211	3 3
Pop	0	↓	1	↓	2	↓	3	↓	215	3
Pop	3	↓	0	↓	1	↓	2	↓	21	3 1
Pop	2	↓	3	↓	0	↓	1	↓	221	3 3
Pop	1	↓	2	↓	3	↓	0	↓	223	3 5
Pop	0	↓	1	↓	2	↓	3	1111	225	3
Push	1	0000	2	0000	3	0000	0	0000	231	403
Push	2	0000	3	0000	0	1111	1	0000	23	411
Pop	1	↓	2	↓	3	↓	0	↓	243	415
Pop	0	↓	1	↓	2	↓	3	↓	245	41
Pop	3	↓	0	↓	1	↓	2	↓	24	421
Pop	2	↓	3	↓	0	↓	1	↓	251	423
Pop	1	↓	2	↓	3	1111	0	↓	253	425
Push	2	0000	3	0000	0	0000	1	0000	25	431
Push	3	0000	0	1111	1	0000	2	0000	2 5	43
Pop	2	↓	3	↓	0	↓	1	↓	2 1	443
Pop	1	↓	2	↓	3	↓	0	↓	2 3	445
Pop	0	↓	1	↓	2	↓	3	↓	2 5	44
Pop	3	↓	0	↓	1	↓	2	↓	2	451
Pop	2	↓	3	1111	0	↓	1	↓	301	453
Push	3	0000	0	0000	1	0000	2	0000	305	45

* Replace 1's with 0's and 0's with 1's.

Note: w_0 is word 0, w_1 is word 1, etc.

Table 5-4 Stack Contents to Check Word Independence

Operation	Stk	w_0	Stk	w_1	Stk	w_2	Stk	w_3	Vector
Push	0	0101	1	1111	2	1111	3	1111	465
Push	1	0101	2	1111	3	1111	0	0101	4[illegible]
Pop	0	0101	1	1111	2	1111	3	0101	4 1
Push	1	0101	2	1111	3	1111	0	0101	4 3
Push	2	0101	3	1111	0	0101	1	0101	4 5
Pop	1	0101	2	1111	3	0101	0	0101	4[illegible]
Push	2	0101	3	1111	0	0101	1	0101	501
Push	3	0101	0	0101	1	0101	2	0101	503
Pop	2	0101	3	0101	0	0101	1	0101	505
Push	3	0101	0	0101	1	0101	2	0101	50[illegible]
Push	0	0101	1	0101	2	0101	3	0101	511
Pop	3	0101	0	0101	1	0101	2	0101	513

Note 1. On Push, old Stk 3 word changes to the new word being pushed. Other Stk contents remain the same.

2. Stk 0 contents on vectors with underlined words are observed on the Y outputs to verify that 0101 was stored in that word location.

Table 5-5 *Stack Contents to Check Bit Independence*

b) It was found that the original test did not verify IC pin and data path independence. During the test, the OR inputs were either all ones or all zeroes. Also, the even and odd bits of the R inputs and the even and odd bits of the D inputs were always in the same state. The addition of the switching speed test patterns corrected these deficiencies. The independence of $\overline{OE}$ to the other pins was only partially checked because $\overline{OE}$ was held at a zero during the entire test. The capability of the device to go into the high-impedence state was never verified. Vectors were added to complete the test for independence and to verify the high-impedence capability.

c) The bit independence and integrity of the instruction, stack, and program counter registers were verified. All flip-flops were checked for 0 to 0, 0 to 1, 1 to 0, and 1 to 1 transitions. The independence of each line in the five

data paths (Instruction Register to Y, Stack Outputs to Y, OR inputs to Y, D inputs to Y, and Program Counter to Stack) was verified and it was shown that each line can pass a one and a zero.

Summary of the Switching Speed Test Evaluation

Vendor A's switching speed tests were divided into propagation delay, set-up time, hold-time, and clock parameter tests. Most of the test patterns were acceptable. However, two of the set-up time patterns specified ones for the OR inputs, thus preventing any failures from being detected. Changes were made to the vectors to correct this problem. Vendor A requested a change of the test mode for the set-up time tests which impacted the evaluation. This request, made after the initial evaluation of the patterns, necessitated another evaluation to determine if the tests were still valid. Vendor A does not presently perform 100 percent hold-time tests on these devices, but they did supply GE with recommended tests which were reviewed, augmented, and included in the slash sheet. In the clock parameter tests the minimum clock high and low times were verified using the functional test patterns. A maximum frequency test was not performed but one was added by GE.

Discussion

The propagation delay, set-up time, and hold time tests were performed with a number of short patterns. GE's tester could not be used to extract these patterns from the test program that Vendor A provided on magnetic tape due to the format that was used. Therefore, the vectors were manually reformatted to facilitate the evaluation. The Y outputs which were irrelevant (don't cares) were listed as zeroes in Vendor A's test patterns even though some of them were actually ones. To eliminate confusion, all the irrelevant Y outputs were changed to X's. Vendor A did not provide any switching speed test patterns used exclusively to check the clock parameters.

A description of each of the four types of switching speed tests is given below. Any changes that were made to the patterns are also given.

1. Propagation Delay Tests

The tests which verified the maximum propagation delays were well written and very thorough. The delays from the D inputs to the Y and C_{N+4} outputs were checked by having each input assume a one and zero state with the other three in the complement state. The same approach was used to check the delays from the OR inputs to the Y and C_{N+4} outputs and the delays from the clock to the Y and C_{N+4} outputs via the R registers. The delay from the clock to the Y and C_{N+4} outputs via the program counter was checked by forcing the incrementer to count from 0 to 15 and observing the Y and C_{N+4} outputs at the specified propagation delay. The delays from the clock to the Y and C_{N+4} outputs via the stack were tested by pushing four groups of words into the stack, one at a time, and verifying the delay to pop them out. The four groups of words were as follows:

1) 0001, 0010, 0100, 1000
2) 1110, 1101, 1011, 0111
3) 0000, 1111, 0000, 1111
4) 1111, 0000, 1111, 0000

In order to verify the delay from C_N to C_{N+4}, 1111 was applied to the incrementer and C_N was forced to go high and low. The delay from C_N to C_{N+4} was specified as 18 ns by Vendor A. The differences between the delays from other inputs to Y and other inputs to C_{N+4} were 10 to 12 ns even though the same number of gates are in each path. Vendor A said that the input transistor on the C_N input is larger than the internal transistors and that the higher capacitance of this transistor causes the C_N to C_{N+4} delay to be longer.

The delay from S0 and S1 to Y and C_{N+4} was thoroughly checked by applying the 12 possible transitions of S0 and S1 and by sensitizing the Y and C_{N+4} outputs to detect any failures. A list of these 12 transitions and the part of the multiplexer that is being tested by each vector are given in Tables 5-6 and 5-7, respectively.

	Transitions		No Change	Vector(s) Which Tests This Combination
1.	S_0	$0 \rightarrow 1$	($S_1 = 1$)	710
2.	S_0	$1 \rightarrow 0$	($S_1 = 1$)	711
3.	S_0	$0 \rightarrow 1$	($S_1 = 0$)	714
4.	S_0	$1 \rightarrow 0$	($S_1 = 0$)	715
5.	S_1	$0 \rightarrow 1$	($S_0 = 1$)	722
6.	S_1	$1 \rightarrow 0$	($S_0 = 1$)	721 & 723
7.	S_1	$0 \rightarrow 1$	($S_0 = 0$)	705
8.	S_1	$1 \rightarrow 0$	($S_0 = 0$)	704
*9.	S_0	$0 \rightarrow 1$		716 & 720
	S_1	$0 \rightarrow 1$		
*10.	S_0	$1 \rightarrow 0$		717
	S_1	$1 \rightarrow 0$		
*11.	S_0	$0 \rightarrow 1$		706
	S_1	$1 \rightarrow 0$		
*12.	S_0	$1 \rightarrow 0$		707
	S_1	$0 \rightarrow 1$		

* The combinations in which transitions occur on both S_0 and S_1 are not required but were performed by Vendor A.

Table 5-6 Twelve Combinations of S_0 and S_1

The pattern submitted by Vendor A for the S0 and S1 tests listed only half of the vectors actually used. The other half were generated by having the test program invert the data on the multiplexer inputs and the Y outputs. GE expanded the pattern to include these vectors.

To test the delay for the $\overline{ZERO}$ input, the Y and C_{N+4} outputs were forced high by driving the multiplexer with all ones. Then the $\overline{ZERO}$ input was driven low and the outputs were checked. A list of the multiplexer inputs that are tested by each vector is given in Table 5-8.

The delay to enable and disable the three-state outputs was verified with both ones and zeroes on the Y outputs.

2. Set-Up Time Tests

Vendor A originally ran the set-up time patterns twice using the return-to-zero (RZ) and return-to-one (RO) test modes.

Vector	S1, S0	Gate Inputs PC 1 2 3	Stack 1 2 3	IR 1 2 3	Direct 1 2 3	Y	
04	1 1	0 0	1 0	0 1	0 1 1	0000	*
05	1 1	0 0	1 0	0 1	0 1 1	0000	
06	1 1	0 0	1 0	0 1	0 1 1	0000	
07	1 1	0 0	1 0	0 1	0 1 1	0000	
00	1 1	0 0	1 0	0 1	0 1 1	0000	
01	0 1	1 0	0 0	0 1 1	0 1	0000	
02	1 1	0 0	1 0	0 1	1 1 1	1111	
03	1 0	0 1	[0 1 1]	0 0	1 0	0000	
04	0 0	[1 1 1]	0 1	1 0	0 0	1111	
05	1 0	0 1	[0 1 1]	0 0	1 0	0000	
06	0 1	1 0	0 0	[1 1 1]	0 1	1111	
07	1 0	0 1	[0 1 1]	0 0	1 0	0000	
10	1 1	0 0	1 0	0 1	[1 1 1]	1111	
11	1 0	0 1	[0 1 1]	0 0	1 0	0000	
12	1 1	0 0	1 0	0 1	0 1 1	0000	*
13	0 0	[0 1 1]	0 1	1 0	0 0	0000	
14	0 1	1 0	0 0	[1 1 1]	0 1	1111	
15	0 0	[0 1 1]	0 1	1 0	0 0	0000	
16	1 1	0 0	1 0	0 1	[1 1 1]	1111	
17	0 0	[0 1 1]	0 1	1 0	0 0	0000	
20	1 1	0 0	1 0	0 1	[1 1 1]	1111	
21	0 1	1 0	0 0	[0 1 1]	0 1	0000	
22	1 1	0 0	1 0	0 1	[1 1 1]	1111	
23	0 1	1 0	0 0	[0 1 1]	0 1	0000	

Notes:

1. The inputs that are labelled 1 are the data inputs of the AND gates in the multiplexer and are only specified for the input that is selected. The inputs that are labelled 2 and 3 are the control inputs of the AND gates and are determined by S0 and S1.

2. The test is repeated with the complement of the data on input 1.

3. Only the vectors with boxes ([ ]) around the AND inputs perform valid propagation delay tests.

4. Asterisk * denotes set-up vectors.

Table 5-7 Control and Data Inputs to the Multiplexer During the S_0 and S_1 Propagation Delay Tests

Vector	$\overline{Zero}$	Gate Inputs					Y	C_{N+4}
		PC	Stk	IR	Direct	OR		
		1 2 3	1 2 3	1 2 3	1 2 3	1 2		
24	1				1 1 1		1111	0
25	1	1 1 1					1111	1
26	0	[1 1 1]					0000	0
27	1	[1 1 1]					1111	1
30	1				1 1 1		1111	0
31	1				1 1 1		1111	0
32	1				1 1 1		1111	0
33	1				1 1 1		1111	0
34	1		1 1 1				1111	1
35	0		[1 1 1]				0000	0
36	1		[1 1 1]				1111	1
37	1				1 1 1		1111	1
40	1			1 1 1			1111	1
41	0			[1 1 1]			0000	0
42	1			[1 1 1]			1111	1
43	1				1 1 1		1111	1
44	0				[1 1 1]		0000	0
45	1				[1 1 1]		1111	1
46	1					1 1	1111	1
47	0					[0 1]	0000	0
50	1					[1 1]	1111	1

Notes: 1. The inputs that are labelled 1 are the data inputs of the AND gates in the multiplexer and are only specified for the input that is selected. The inputs that are labelled 2 and 3 are the control inputs of the AND gates and are determined by S0, S1, and $\overline{ZERO}$.

2. Only the vectors with boxes ([ ]) around the AND inputs perform valid propagation delay tests.

Table 5-8 Multiplexer Inputs During the $\overline{ZERO}$ Propagation Delay Tests

After GE had completed their evaluation of these tests, Vendor A requested to use their tester's exclusive-or (XOR) mode for the input under test since this would facilitate calibration of their tester. An evaluation revealed that the change in test mode had a significant impact on the validity of some of the tests since more than one input was being tested at a time. Vendor A then suggested changing the test so that only one input would be tested at a time. The affected patterns were reviewed again to ensure that all of the set-up times were being verified.

There are three parts of the 2909 for which set-up times are specified: the instruction registers, the program counter registers, and the stack pointer. The evaluation of the tests for each part is discussed below.

a) Instruction Registers

The set-up time for the Register Enable input ($\overline{RE}$) was verified by forcing it low and high at the specified set-up time before the rising edge of the clock and checking whether the contents of the R registers changed or remained the same. A "1 to 0" and "0 to 1" transition was made for each bit before the clock. The change in test mode did not impact these tests.

b) Program Counter Registers

The set-up time tests for the program counter registers were more difficult to evaluate because the vectors were not in a logical sequence. The change in test mode had a significant impact on thest tests.

Checking of these set-up times required a two vector sequence. The input under test was changed at the specified set-up time before the clock on the first vector. On the following vector, the data that should have been loaded into the program counter was verified at the Y outputs. The tests using the RZ and RO modes and the XOR mode testing four inputs at once were not complete.

Table 5-9 indicates which transitions could not be verified using these test modes. The tests in which only one pin is tested at a time using the XOR mode are complete. No additional vectors were required.

Some modifications were made to the pattern which checks the OR inputs. The OR inputs were logic 1's on many of the vectors when the data from the program counter should have been verified and any set-up time failure would be masked. Therefore, all of the OR inputs were changed to zeroes on these vectors.

The C_N input was completely tested by the occurrence of a 0 to 1 and 1 to 0 transition on each data path from the C_N input to the program counter registers. One pattern is used to test the S0, S1, and $\overline{ZERO}$ inputs. The set-up times are verified for all eight combinations of these three inputs. A few vectors had to be modified, however, because the OR inputs were again ones when the program counter output was being verified.

c) Stack Pointer

The tests which verify the set-up times for the Push/Pop input (PUP) and the File Enable input ($\overline{FE}$) did not require any changes or additions. The failures which could occur if the set-up times are not met are listed in Tables 5-10 and 5-11. All possible failures are detected for each combination of PUP, $\overline{FE}$, and the logic states of the flip-flops in the stack pointer. A list of the operation being verified by each vector and the failures which can be detected for the PUP tests are given in Table 5-12.

3. Hold Time Tests

Vendor A does not presently perform hold time tests on each device but did provide GE with a set of vectors which could be used for this purpose. The vectors submitted by Vendor A which checked the hold times for $\overline{RE}$, R_i, $\overline{FE}$, and PUP were identical to the ones for the set-up times on these

		D_3, OR_3				D_2, OR_2				D_1, OR_1				D_0, OR_0			
Carry-In (C_{IN})		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Transition		0 to 1	0 to 1	1 to 0	1 to 0	0 to 1	0 to 1	1 to 0	1 to 0	0 to 1	0 to 1	1 to 0	1 to 0	0 to 1	0 to 1	1 to 0	1 to 0
Original Test	Invert RT Mode		X	X			X	X			X	X			X	X	
	Normal RZ Mode	X	X			X	X			X	X			X			X
Modified Test	XOR Mode (up to 4 inputs)	X	X	X	X	X	X	X	X	X		X		X	X	X	X
	XOR Mode (only 1 input)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: An X under a particular bit indicates that this transition is checked during the test.

Table 5-9 Transitions Which Are Checked During the D and OR Set-Up Time for Various Test Modes

Operation Being Performed	Possible Failures
POP (PUP = 0) -- Counter Decrements	POP --- Counter Increments PUSH -- Counter Increments PUSH -- Counter Decrements
PUSH (PUP = 1) -- Counter Increments	PUSH -- Counter Decrements POP --- Counter Decrements POP --- Counter Increments

Table 5-10 PUP Set-Up Time Tests

Operation Being Performed	Possible Failures
No Operation ($\overline{FE}$ = 1) or PUSH ($\overline{FE}$ = 0, PUP = 1) or POP ($\overline{FE}$ = 0, PUP = 0)	PUSH -- Increment * PUSH -- Decrement PUSH -- No Change PUSH -- Increment Twice POP --- Increment POP --- Decrement * POP --- No Change POP --- Increment Twice

* Normal Operation for PUSH and POP

Table 5-11 $\overline{FE}$ Set-Up Time Tests

Vector	Operation to be Performed	Expected Y Output	Possible Failures	Y Output for each Failure	Validity of Test
13[illegible]	POP-D	0110	POP-I	xxxx	FND
			PUSH-I	0110	FND
			PUSH-D	0110	FND
13[illegible]	POP-D	1011	POP-I	1011	FND
			PUSH-I	1100	FD
			PUSH-D	1100	FD
13[illegible]	PUSH-I	0111	PUSH-D	0111	FND
			POP-D	xxxx	FDSV
			POP-I	0111	FDSV
13[illegible]	PUSH-I	1100	PUSH-D	1100	FND
			POP-D	1011	FD
			POP-I	1011	FD
13[illegible]	PUSH-I	1000	PUSH-D	1000	FDSV
			POP-D	0111	FD
			POP-I	xxxx	FND
13[illegible]	POP-D	1100	POP-I	1011	FD
			PUSH-I	1101	FD
			PUSH-D	1101	FD
13[illegible]	POP-D	0111	POP-I	1000	FD
			PUSH-I	1001	FD
			PUSH-D	1001	FD
13[illegible]	PUSH-I	1101	PUSH-D	1101	FND
			POP-D	1011	FD
			POP-I	1100	FD
13[illegible]	PUSH-I	1000	PUSH-D	1000	FDSV
			POP-D	0111	FD
			POP-I	1000	FND
13[illegible]	PUSH-I	1110	PUSH-D	1110	FDSV
			POP-D	1101	FD
			POP-I	1011	FD
13[illegible]	POP-D	1000	POP-I	0111	FD
			PUSH-I	1001	FD
			PUSH-D	1001	FD
13[illegible]	POP-D	1101	POP-I	1110	FD
			PUSH-I	1111	FD
			PUSH-D	1111	FD

Table 5-12 Operations Performed by the PUP Set-Up Time Tests

Vector	Operation to be Performed	Expected Y Output	Possible Failures	Y Output for each Failure	Validity of Test
1401	PUSH-I	1001	PUSH-D	1001	FDSV
			POP-D	0111	FD
			POP-I	1000	FD
1402	PUSH-I	1110	PUSH-D	1110	FDSV
			POP-D	1101	FD
			POP-I	1110	FSD
1403	PUSH-I	1010	PUSH-D	1010	FDSV
			POP-D	1001	FD
			POP-I	0111	FD
1404	POP-D	1110	POP-I	1101	FD
			PUSH-I	1111	FD
			PUSH-D	1111	FD
1405	POP-D	1001	POP-I	1010	FD
			PUSH-I	1011	FD
			PUSH-D	1011	FD
1406	POP-D	1101	POP-I	1110	FD
			PUSH-I	1111	FD
			PUSH-D	1111	FD
1407	POP-D	1010	POP-I	1001	FD
			PUSH-I	1010	FDSV
			PUSH-D	1010	FSD
1410	POP-D	1110	POP-I	1101	FD
			PUSH-I	1110	FDSV
			PUSH-D	1110	FSD
1411	POP-D	1001	POP-I	1010	FD
			PUSH-I	1011	FD
			PUSH-D	1011	FD
1412	PUSH-I	1111	PUSH-D	1111	FSD
			POP-D	1101	FD
			POP-I	1110	FD
1413	PUSH-I	1010	PUSH-D	1010	FDSV
			POP-D	1001	FD
			POP-I	1010	FSD
1414	PUSH-I	0000	PUSH-D	0000	FDSV
			POP-D	1111	FD
			POP-I	1111	FD

Table 5-12 (cont'd) Operations Performed by the PUP Set-Up Time Tests

Vector	Operation to be Performed	Expected Y Output	Possible Failures	Y Output for each Failure	Validity of Test
1415	POP-D	1010	POP-I	1001	FD
			PUSH-I	1011	FD
			PUSH-D	1011	FD
1416	POP-D	1111	POP-I	0000	FD
			PUSH-I	0001	FD
			PUSH-D	0001	FD

Notes: 1. D is decrement
I is increment
X is data unknown

2. Definitions of Validity of Test:
FD -- Failure will be detected from Y output
FDNV -- Failure will be detected by a successive vector
FND -- Failure will not be detected

Table 5-12 (Concluded) Operations Performed by PUP Set-Up Time Tests

inputs and did not require any modification. The input under test data in the hold time test patterns for the D_i, OR_i, C_N, S1, S0, and $\overline{ZERO}$ inputs was inverted from what it was in the set-up time patterns. The Y outputs were different because they were observed before the clock low to high edge in order to properly check the hold times for the data which enters the Program Counter. The tests for these six inputs were found to be inadequate. Instead of modifying these tests, a decision was made to use the vectors from the set-up time tests. A few changes were required to the Y outputs for the C_N, S1, S0, and $\overline{ZERO}$ tests because the outputs are checked before the clock pulse.

4. Clock Parameter Tests

Vendor A used the functional test patterns to verify the minimum clock high and low times. However, they did not specify and verify a maximum frequency. GE learned from Vendor A that the period for the maximum frequency is greater than the sum of the minimum clock high and low times. Therefore, a maximum frequency test was added by GE to verify this parameter. The test uses the functional test patterns and specifies a limit obtained from Vendor A.

SECTION VI

TEST DEVELOPMENT FOR THE 2910 MICROPROGRAM CONTROLLER

Objective

The purpose of this evaluation was to review the tests which Vendor A had submitted for inclusion in the slash sheet for the 2910. The approach used was the Procedure for LSI Test Development which is documented in RADC Report RADC-TR-78-138 and will not be repeated here. The block diagram for the 2910 is shown in Figure 6-1.

Summary of the Functional Test Evaluation

Vendor A provided GE with a gate-level logic diagram and a description and listing of their functional and switching speed test patterns. Vendor A's functional test consists of 2079 vectors which are divided into the following nine test patterns: a preliminary check, a pattern for each of five functional blocks, and three switching speed patterns. The 2910 is thoroughly tested by these patterns except for bit independence within the word locations in the Stack and the Y output High-Impedance capability. GE added vectors to correct these deficiencies.

Discussion

For test purposes Vendor A divided the block diagram in Figure 6-1 into the following five functional blocks:

1. Multiplexer
2. Register/Counter and Zero Detector
3. Incrementer and Microprogram Counter/Register
4. Stack and Stack Pointer
5. Instruction PLA

The Procedure for LSI Test Development was used as a guideline for determining the test requirements for each of these blocks.

Vendor A provided GE with a description and listing of their functional and switching speed test patterns and a gate-level logic diagram. The functional test consists of a preliminary check pattern, a pattern for each of the five functional blocks, and three switching speed patterns. All of the functional test patterns are performed with an early strobe in which the outputs are looked at before the rising edge of the clock. Therefore, any register contents which change on the rising edge of the clock must be verified on the following vector. Each block was completely tested except the Stack and Stack Pointer, for which bit independence in each word was not thoroughly checked. A brief description of each pattern and any changes

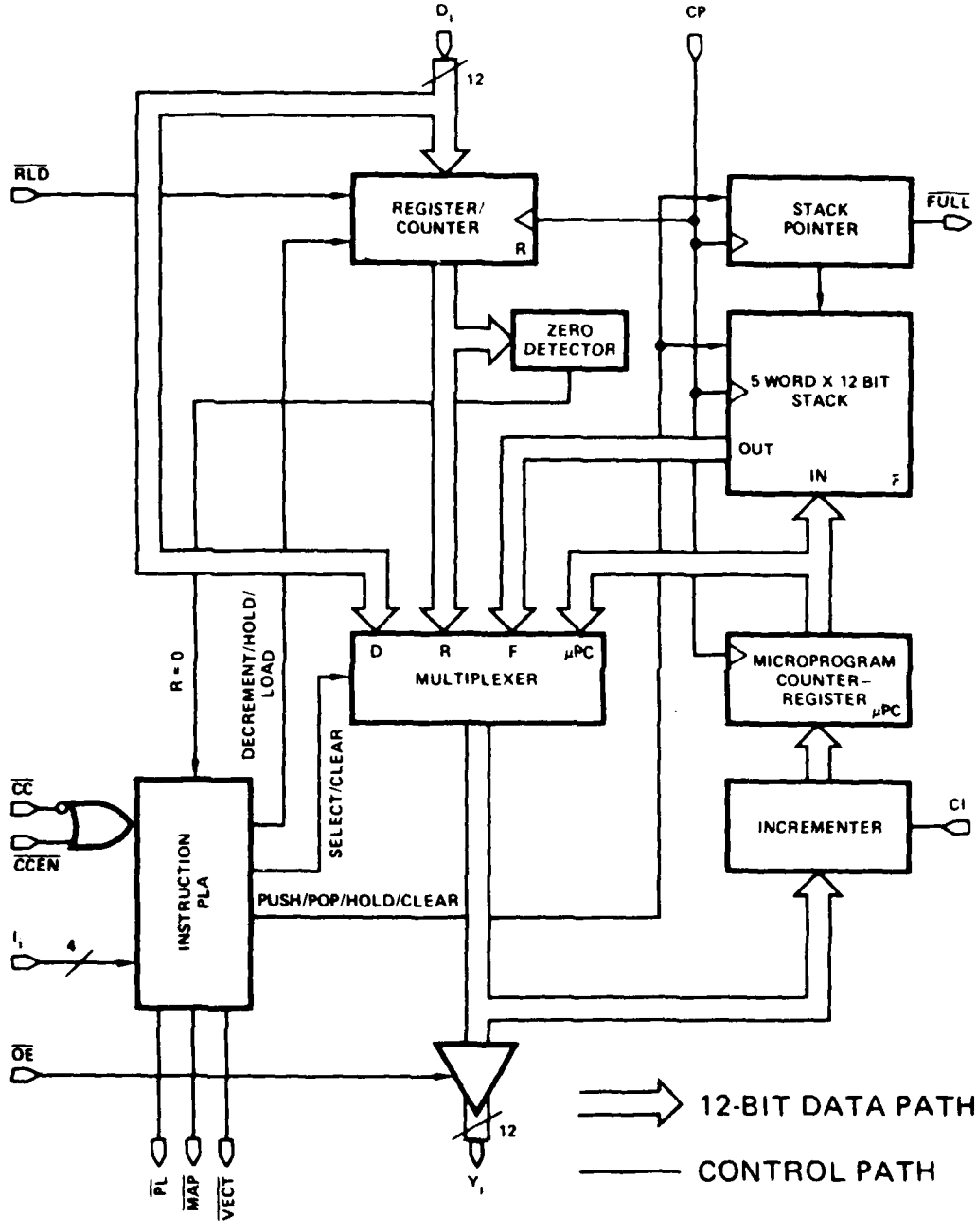


Figure 6-1 2910 BLOCK DIAGRAM

that were recommended by us are given below.

1. Preliminary Check

This pattern of vectors is used for an initial check of the [illegible] before the remainder of the functional test is performed. Data is loaded into the three sets of registers (Program Counter, Register Counter and Stack) and the contents of each are verified. Data is read from each of the four Multiplexer data sources. This pattern is very effective in verifying the basic operation of each of the functional blocks.

2. Multiplexer

The Multiplexer pattern checks the capability to pass a one and a zero in each selection position. The pattern is designed to have two data words (AAA and 555) pass from each data source while the complemented word (555 and AAA) is at the other three sources. However, Register/Counter inputs were not the complement of the Stack inputs when the Stack was being used as the data source. To load the complement into the Register/Counter GE changed the SRC input to logic "0" on vectors 32, 40, 1203, and 1208. Table 6-1 shows the input states for each data source during the test including the change which GE made.

The RESET instruction on the Multiplexer is verified by applying ones to the data inputs during RESET and verifying that zeroes are present on the Y outputs. Bit independence of the data inputs is not completely verified by this pattern. However, it is completely when the patterns for the other functional blocks are used.

3. Register/Counter and Zero Detector

The Register/Counter and Zero Detector are thoroughly tested by this pattern. The first part of the test loads 111, 000, 555, and AAA into the Register/Counter and then verifies the presence of data while the complemented data is on the D inputs. The remainder of the pattern loads a one and then a zero into each bit of the register with the complement loaded into the other eleven. A flow chart which shows what the subroutine does is given in Figure 6-2. The Zero Detector checks the contents of the Register/Counter for zero thus determining whether a hold or decrement operation is performed. The Zero Detector output to the PLA is verified for both zero and non-zero data in the Register/Counter. It is verified that the RLD input takes priority over the load, hold, and decrement control inputs from the PLA. Bit independence and register integrity are checked for the Register/Counter.

4. Incrementer and Microprogram Counter

This pattern is divided into two subroutines which thoroughly test

Selected Input	Data at Selected Input	Data at Deselected Inputs			
		D	R	F	PC
D	555 AAA		AAA 555	AAA 555	AAA 555
R	555 AAA	AAA 555		AAA 555	AAA 555
F	555 AAA	AAA 555	555[2] AAA[3]		AAA 555
PC	555 AAA	AAA 555	AAA 555	AAA 555	
RESET	000	FFF	FFF	FFF	FFF

Notes: 1) D = D Inputs, R = Register/Counter, F = Stack, PC = Program Counter
2) Changed to AAA
3) Changed to 555

Table 6-1 Input States (Hexadecimal) of Multiplexer Data Sources

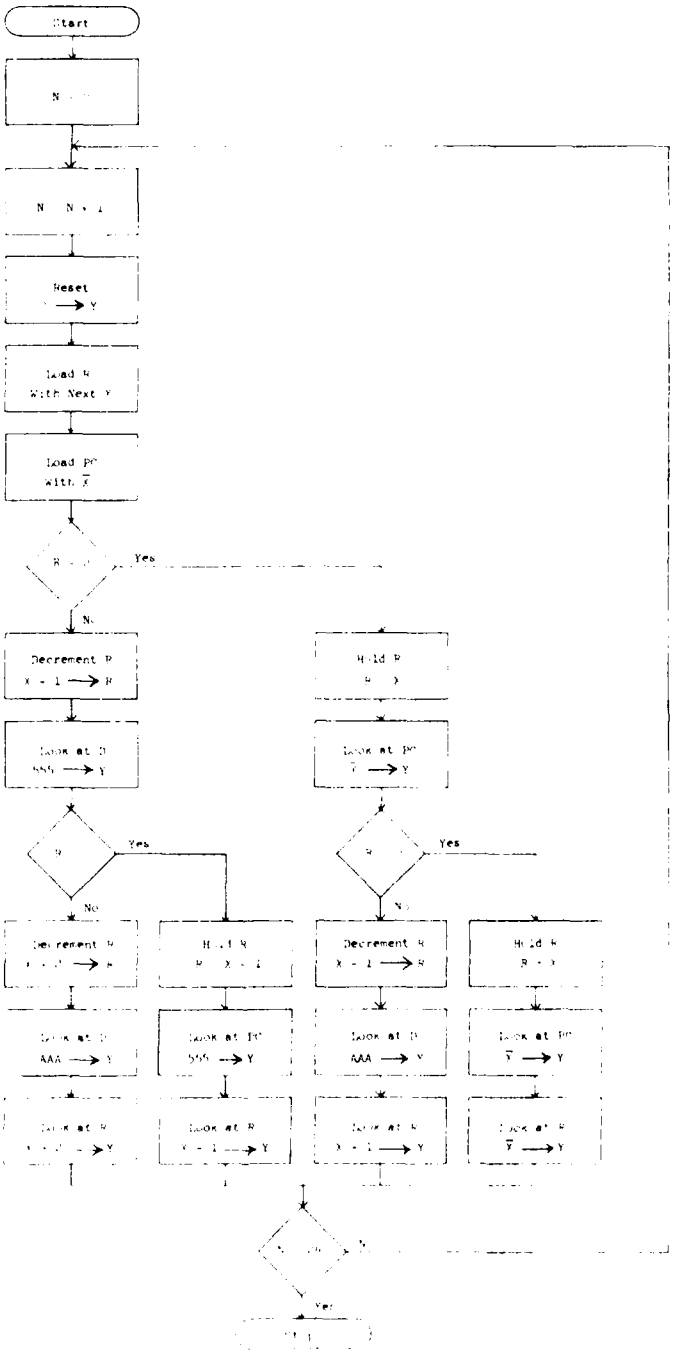


Figure 6-2 Flowchart for the Register/Counter Subroutine

the Incrementer and Program Counter for stuck-at-faults. The first subroutine consists of a two vector sequence which is repeated 13 times. The first vector loads the following data words into the Program Counter while the CI input is at a logic "0" and the second vector verifies them at the Y outputs:

000000000000
000000000001
000000000011
000000000111
000000001111
000000011111
000000111111
000001111111
000011111111
000111111111
001111111111
011111111111
111111111111

The second subroutine contains three vectors which are repeated 13 times. The first vector of this subroutine also loads the above data words into the Program Counter while the CI input is at a logic "0". However, on the next vector the CI input is at a logic "1" resulting in the following data words being incremented before they are loaded into the Program Counter:

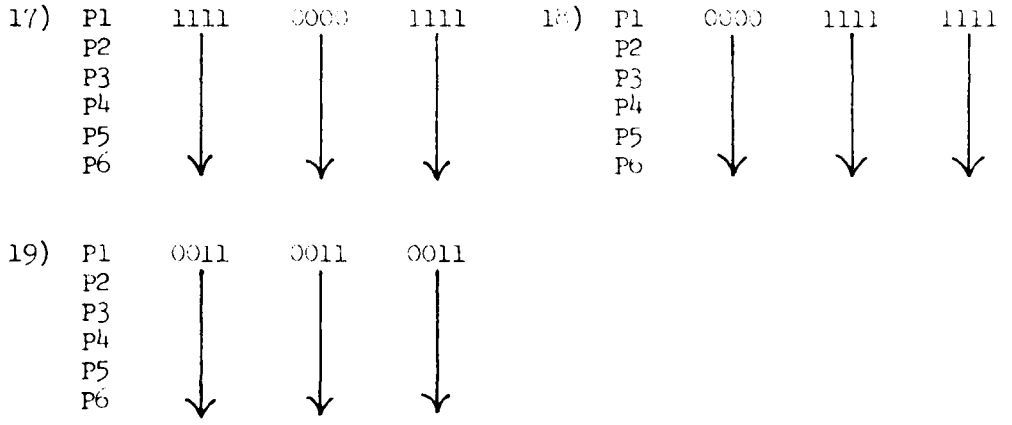
111111111110
111111111101
111111111011
111111110111
111111101111
111111011111
111110111111
111101111111
111011111111
110111111111
101111111111
011111111111
111111111111

The incremented data words are the complement of the initial data words that were loaded into the Program Counter. By verifying the contents of the Program Counter on the last vector, the test for the carry-in circuitry is completed. Bit independence and register integrity are not checked completely by this pattern. However, these tests are completed by the Stack and Stack Pointer pattern which uses the Program Counter to input data into the Stack.

5. Stack and Stack Pointer

The Stack and Stack Pointer are tested with the subroutine shown in Table 6-2. The following operations are performed: a clear, a hold, five pushes, and eight pops. The 16 groups of words shown in Table 6-3 are pushed into the Stack and are verified at the Y outputs when they are popped out. Each group consists of six words but only the first five (P_1 - P_5) actually get written into the Stack. The sixth one could be written into the Stack if a failure occurs. The Stack output for the three extra pops is not obvious but is predictable. For all three the Stack output is the third word (P_3) that is written into the Stack.

This pattern thoroughly checks word independence and register integrity but does not completely check bit independence within each word. Figure 6-3 shows which bits were not completely checked. The results are only given for four Stack locations: 00, 01, 10, and 11. The fifth location is the Stack register through which all Stack data passes. It will be thoroughly tested if the other locations are thoroughly tested. The following three groups of words were added to the test by GE to complete the test for bit independence:



This pattern also includes an "extra push" test in which six consecutive pushes are performed. Vendor A's data sheet states that the word at the top of the Stack is written over but it is actually the second word down (the fourth word written into the Stack) which is lost. Table 6-4 shows the Stack contents and Y outputs for each vector for the extra push test. Since five or more pushes are performed during both parts of the Stack test, the operation of the FULL output is also verified.

6. Instruction PLA

The Instruction PLA is thoroughly tested by this pattern which contains 1811 vectors. The pattern contains a subroutine which is repeated 10 times. Each of the 16 instructions is performed for all

Instr	Stack Oper.	Stack Pointer	PC	Stack Locations					Y^-	Y^+
				S.R.	00	01	10	11		
0	C	000	000	U	U	U	U	U	000	000
2	H	000	P_1	U					P_1	P_1
1	U	001	P_2	P_1			↓		P_2	P_2
1	U	010	P_3	P_2			P_1	↓	P_3	P_3
1	U	011	P_4	P_3	↓			P_2	P_4	P_4
1	U	100	P_5	P_4	P_3	↓			P_5	P_5
1	U	101	P_6	P_5		P_4			P_6	P_6
A	0	100	P_5	P_4					P_5	P_4
A	0	011	P_4	P_3					P_4	P_3
A	0	010	P_3	P_2					P_3	P_2
A	0	001	P_2	P_1					P_2	P_1
A	0	000	P_1	P_4					P_1	P_4
A	0	000	P_4	P_3					P_4	P_3
A	0	000	P_3	P_3					P_3	P_3
A	0	000	P_3	P_3	↓	↓	↓	↓	P_3	P_3

Notes: 1) Under Stack Operation, C = Clear, H = Hold, U = Push, P = Pop

2) S.R. = Stack Register, PC = Program Counter

3) Y^- = Y outputs with early strobe

4) Y^+ = Y outputs with late strobe

5) U = Undefined for Stack locations

6) Data is in Hexadecimal

Table 6-2 Stack and Stack Pointer Subroutine #1

1)		2)		3)		4)	
P_1	[illegible]	P_1	[illegible]	P_1	AAA	P_1	[illegible]
P_2	[illegible]	P_2	[illegible]	P_2	[illegible]	P_2	AAA
P_3	[illegible]	P_3	[illegible]	P_3	AAA	P_3	[illegible]
P_4	[illegible]	P_4	[illegible]	P_4	555	P_4	AAA
P_5	[illegible]	P_5	[illegible]	P_5	AAA	P_5	[illegible]
P_6	[illegible]	P_6	FBE	P_6	555	P_6	AAA
5)		6)		7)		8)	
P_1	FFF	P_1	[illegible]	P_1	[illegible]	P_1	FFF
P_2	000	P_2	FFF	P_2	FFF	P_2	[illegible]
P_3	000	P_3	FFF	P_3	[illegible]	P_3	FFF
P_4	000	P_4	FFF	P_4	[illegible]	P_4	FFF
P_5	000	P_5	FFF	P_5	000	P_6	FFF
P_6	AAA	P_6	AAA	P_6	AAA	P_7	AAA
9)		10)		11)		12)	
P_1	000	P_1	FFF	P_1	[illegible]	P_1	FFF
P_2	000	P_2	FFF	P_2	[illegible]	P_2	FFF
P_3	FFF	P_3	000	P_3	[illegible]	P_3	FFF
P_4	000	P_4	FFF	P_4	FFF	P_4	000
P_5	000	P_5	FFF	P_5	000	P_6	FFF
P_6	AAA	P_6	AAA	P_6	AAA	P_7	AAA
13)		14)		15)		16)	
P_1	[illegible]	P_1	FFF	P_1	[illegible]	P_1	FFF
P_2	000	P_2	FFF	P_2	[illegible]	P_2	FFF
P_3	[illegible]	P_3	FFF	P_3	[illegible]	P_3	FFF
P_4	[illegible]	P_4	FFF	P_4	[illegible]	P_4	FFF
P_5	FFF	P_5	000	P_5	[illegible]	P_6	FFF
P_6	AAA	P_6	AAA	P_6	AAA	P_7	AAA

Table 6-3 Patterns For Stack and Stack Pointer (Hexadecimal)

Note: An X indicates that bit independence has been checked for the two corresponding bits for one of the two required conditions.

Stack Location 00

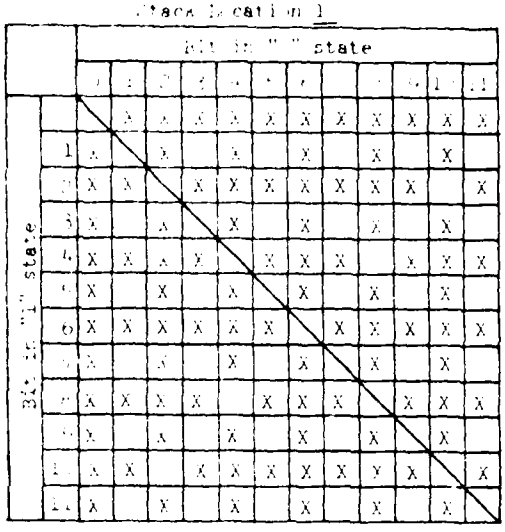
Bit in "1" state \ Bit in "0" state	0	1	2	3	4	5	6	7	8	9	10	11
0		X	X	X		X		X	X	X		X
1	X		X	X	X		X	X	X		X	X
2	X	X		X	X	X	X	X		X	X	X
3	X	X	X		X	X	X		X	X	X	
4		X	X	X		X		X	X	X		X
5	X		X	X	X		X	X	X		X	X
6		X	X	X		X		X	X	X		X
7	X	X	X		X	X	X		X	X	X	
8	X	X		X	X	X	X	X		X	X	X
9	X		X	X	X		X	X	X		X	X
10		X	X	X		X		X	X	X		X
11	X	X	X		X	X	X		X	X	X	

Stack Location 01

Bit in "1" state \ Bit in "0" state	0	1	2	3	4	5	6	7	8	9	10	11
0		X	X	X		X	X	X		X	X	X
1	X		X	X	X		X		X	X	X	
2	X	X		X	X	X		X	X	X		X
3	X	X	X		X	X	X	X	X		X	X
4		X	X	X		X	X	X		X	X	X
5	X		X	X	X		X		X	X	X	
6	X	X		X	X	X		X	X	X		X
7	X		X	X	X		X		X	X	X	
8		X	X	X		X	X	X		X	X	X
9	X	X	X		X	X	X	X	X		X	X
10	X	X		X	X	X		X	X	X		X
11	X		X	X	X		X		X	X	X	

Figure 6-3 Results of Bit Independence Evaluation for 4 Stack Locations

Note: An X indicates that bit independence has been checked for the two corresponding bits for one of the two required conditions.



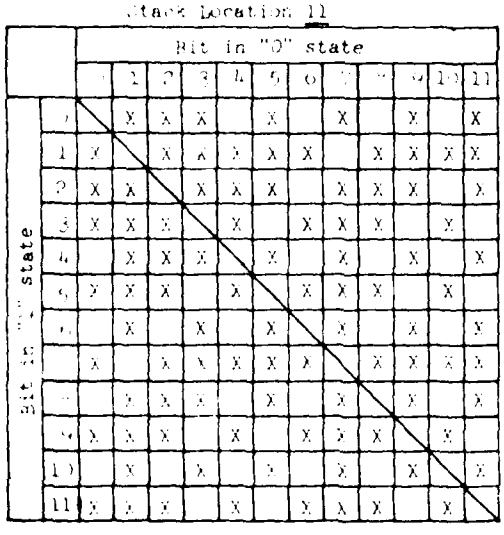


Figure 6-3 Results of Bit Independence Evaluation for a Stack Location (Cont'd.)

Instr.	Stack Oper.		Stack Pointer	P.C.	Stack Locations						MUX Input		Y^-	Y^+	F^-	F^+
					S.R.	H.R.	00	01	10	11						
0	C		000	000	777	U	U	U	U	U	0		000	000	1	1
2	H		000	820	777	U	↓	U	U	U	D		820	820	1	1
1	U		001	410	820	777	↓	777	U	U	D		410	410	1	1
1	U		010	208	410	820	↓	↓	820	U	D		208	208	1	1
1	U		011	104	208	410	↓	↓	↓	410	D		104	104	1	1
1	U		100	082	104	208	208	↓	↓	↓	D		082	082	1	1
1	U		101	041	082	104	↓	104	↓	↓	D		041	041	1	0
1	U		101	AAA	041	082	↓	082	↓	↓	D		AAA	AAA	0	0
A	0		100	041	082	041	↓	↓	↓	↓	F		041	082	0	1
A	0		011	082	208	082	↓	↓	↓	↓	F		082	208	1	1
A	0		010	208	410	208	↓	↓	↓	↓	F		208	410	1	1
A	0		001	410	820	410	↓	↓	↓	↓	F		410	820	1	1
A	0		000	820	082	820	↓	↓	↓	↓	F		820	082	1	1
A	0		000	082	208	082	↓	↓	↓	↓	F		082	208	1	1
A	0		000	208	208	208	↓	↓	↓	↓	F		208	208	1	1
A	0		000	208	208	208	↓	↓	↓	↓	F		208	208	1	1

Notes: 1) Under Stack Operation, C = Clear, H = Hold, U = Push, O = Pop

2) P.C. = Program Counter, S.R. = Stack Register, H.R. = Holding Register

3) Y^- and F^- = Y and F outputs with early strobe

4) Y^+ and F^+ = Y and F outputs with late strobe

5) U = Undefined for Stack locations

6) Under MUX Input, 0 = Reset, D = D Inputs, F = Stack

7) Data is in Hexadecimal

Table 6-4 Stack and Stack Pointer Subroutine #2

combinations of the $\overline{CC}$ input, the $\overline{RLD}$ input, and the Register/Counter contents (000 or 111). When each instruction under test is performed, unique data words are contained in the Stack, Program Counter, and Register/Counter. All 16 outputs (Y_{0-11}, $\overline{PL}$, $\overline{MAP}$, $\overline{VECT}$, and $\overline{FULL}$) are verified for each instruction. It is verified that every operation that should occur on each instruction does occur. If the specified operation is not performed, the failure can be detected and isolated because of the unique data words from each data source. Table 6-5 contains a copy of the subroutine for Instruction 0 with $\overline{CC}$ = 0, $\overline{RLD}$ = 0, and Reg./Cntr. = 000. This particular combination for Instruction 0 verifies all the operations except a Hold of the Register/Counter which is verified by another one of the eight combinations. The subroutines for the other instructions are similar.

7. Switching Speed Patterns

Three of the switching speed test patterns, 1, 2, and 5, are used as part of the functional test. Patterns 1 and 2 use an early strobe. Pattern 1 verifies a group of data words at the Y outputs. The D inputs and the Program Counter are used as data sources. Pattern 2 loads data into the Register/Counter and verifies it at the Y outputs. Pattern 5 does not require a clock. It verifies the three Enable outputs. These three patterns are explained in more detail in the Discussion of the switching speed tests.

General Tests and Comments

Vendor A performs the functional test twice: first, with the inputs switching when the clock is high and then, again, with the inputs switching when the clock is low. Different failures will be detected by each test. The functional test is not performed at maximum frequency and does not verify switching speed parameters. Therefore, separate switching speed tests are performed. Neither the functional nor switching speed tests were performed with the inputs at threshold levels. However, during Vendor A's DC tests threshold levels were applied to some of the inputs as part of the required preconditioning for the V_{OL} and V_{OH} measurements. These DC tests were expanded by GE to apply threshold levels to all the inputs. Any threshold level failure will be detected by a V_{OL} or V_{OH} measurement.

Vendor A's data sheets do not explain what happens to the Stack after a RESET operation or more than five pushes or pops. An explanation of these operations was obtained from Vendor A in order to understand the functional test patterns.

Vendor A's functional test did not contain a pattern to check the high impedance capability of the Y outputs. A pattern to check this was added by GE.

Instruction		$\overline{CC}$	$\overline{CCEN}$	$\overline{RLD}$	$\overline{CI}$		MUX Select	Reg. Oper.	Stack Oper.		Direct Input Data			Program Counter Contents			Stack Pointer	Stack Register Contents			Register/ Counter Contents				Y Outputs				$\overline{PL}$	$\overline{MAP}$	$\overline{VECT}$		$\overline{FULL}$
2		0	0	1	0		D	H	H		A	A	A	A	A	A	XXX	X	X	X	X	X	X		A	A	A	1	0	1		1
3		1	1	1	-		D	H	H		5	5	5	5	5	5	XXX	X	X	X	X	X	X		5	5	5	0	1	1		1
BEGIN SUBROUTINE																																
0		0	0	0	0		0	H	C		1	1	1	0	0	0	000	X	X	X	X	X	X		0	0	0	0	1	1		1
2		0	0	1	0		D	H	H		2	2	2	2	2	2	000	X	X	X	1	1	1		2	2	2	1	0	1		1
1		0	0	1	0		D	H	U		3	3	3	3	3	3	001	2	2	2	1	1	1		3	3	3	0	1	1		1
1		0	0	1	0		D	H	U		4	4	4	4	4	4	010	3	3	3	1	1	1		4	4	4	0	1	1		1
1		0	0	1	0		D	H	U		5	5	5	5	5	5	011	4	4	4	1	1	1		5	5	5	0	1	1		1
1		0	0	1	0		D	H	U		6	6	6	6	6	6	100	5	5	5	1	1	1		6	6	6	0	1	1		1
1		1	1	1	-		D	H	U		7	7	7	7	7	7	101	6	6	6	1	1	1		7	7	7	0	1	1		1
B		0	0	1	0		D	H	O		7	7	7	7	7	7	100	5	5	5	1	1	1		7	7	7	0	1	1		0
B		1	1	1	0		D	H	O		7	7	7	7	7	7	011	4	4	4	1	1	1		7	7	7	0	1	1		1
*0		0	0	0	1		0	H	C		8	8	8	0	0	1	000	7	7	7	8	8	8		0	0	0	0	1	1		1
E		1	1	1	0		PC	H	H		9	9	9	0	0	1	000	7	7	7	8	8	8		0	0	1	0	1	1		1
A		0	0	1	0		F	H	O		9	9	9	7	7	7	000	4	4	4	4	8	8		7	7	7	0	1	1		1
A		0	0	1	0		F	H	O		9	9	9	4	4	4	000	4	4	4	8	8	8		4	4	4	0	1	1		1
C		1	0	1	0		R	H	H		9	9	9	8	8	8	000	4	4	4	8	8	8		8	8	8	0	1	1		1

*Instruction Under Test

→ H = Hold, C = Clear, U = Push, O = Pop

→ H = Hold

→ D = D Inputs, O = Reset, PC = Program Counter, F = Stack, R = Register/Counter

Table 6-5 Subroutine for PLA Pattern on Instruction 0

Summary of the Switching Speed Test Evaluation

The switching speed tests which Vendor A provided checked most of the propagation delays, set-up times, hold times, and clock parameters. Vendor A supplied six switching speed test patterns but many of the functional test patterns are also used to check the switching speed parameters. A High-Impedance pattern was added by GE to check the delay from the $\overline{OE}$ input to the Y outputs. A maximum frequency test was also added by GE. The specified limit was obtained from Vendor A. The pattern which checks the delay for the Register/Counter contents to change from a zero value to a non-zero value is being revised by Vendor A based on GE comments.

Discussion

Vendor A's switching speed tests check propagation delays, set-up times, hold times, and clock parameters. Many of the patterns that are used to measure or verify these parameters are functional test patterns which were previously described. There are also six additional patterns which are used. Table 6-6 shows which patterns are used to check each switching speed parameter. Five of the six switching speed patterns consist of subroutines which use all or part of the following 26 data words:

J	DW(J)	J	$\overline{DW(J)}$
1	000000000000	14	111111111111
2	000000000001	15	111111111110
3	000000000010	16	111111111101
4	000000000100	17	111111111011
5	000000001000	18	111111110111
6	000000010000	19	111111101111
7	000000100000	20	111111011111
8	000001000000	21	111110111111
9	000010000000	22	111101111111
10	000100000000	23	111011111111
11	001000000000	24	110111111111
12	010000000000	25	101111111111
13	100000000000	26	011111111111

An explanation of each of the patterns is given below:

Pattern 1

Pattern 1 is used for checking the D to Y propagation delay, the D set-up time to PC, and the CI set-up time. The following data words are applied to the Multiplexer and Program Counter/Register from the D inputs: DW(J), $\overline{DW(J)}$, and $\overline{DW(J) - 1}$ where DW(J) is defined above for J = 1 - 26. DW(J), $\overline{DW(J)}$, and $\overline{DW(J) - 1}$ are passed by the Multiplexer from the D input and are verified at the Y outputs. DW(J) and DW(J) - 1 are passed by the Multiplexer from the Program Counter/Register input

Note: An X to the right of a Switching Speed Parameter indicates that the pattern listed above the X is used as part of the test which checks that parameter.

Switching Speed Parameters	Strobe	Functional Test Patterns							Switching Speed Test Patterns						
		Preliminary Check	Multiplexer	Register/Counter	Program Counter	Stack	Instruction PLA Normal	Instruction PLA Inverted	1 Normal	1 Inverted	2 Normal	2 Inverted	3	4	4A	5
Combinatorial Delays																
D ⟶ Y	E						X		X		X					
I ⟶ Y	E						X	X	X	X	X	X				
$\overline{CC}$ ⟶ Y	E						X	X	X	X	X	X				
$\overline{CCEN}$ ⟶ Y	E						X	X	X	X	X	X				
I ⟶ Enables	E	X	X	X	X	X										X
Delays From Clock																
CP ⟶ Y (I = 8,9,F)	L													X		
CP ⟶ Y (I = 8,9,F) Special	L														X	
CP ⟶ Y (I = 0-7, A-E)	L				X	X							X			
CP ⟶ FULL	L	X	X	X	X	X							X	X		
Set-Up Times																
D ⟶ Reg	L										X					
D ⟶ PC	L						X		X		X					
I_{0-3}	L						X	X	X	X	X	X				
$\overline{CC}$	L						X	X	X	X	X	X				
$\overline{CCEN}$	L						X	X	X	X	X	X				
CI	L								X							
$\overline{RLD}$	L						X		X		X					
Hold Times																
D ⟶ Reg	E										X					
D ⟶ PC	E								X							
I_{0-3}	E						X		X		X					
$\overline{CC}$	E						X		X		X					
$\overline{CCEN}$	E						X		X		X					
CI	E								X							
$\overline{RLD}$	E						X		X		X					
Clock Parameters																
Minimum Clock Low	E	X	X	X	X	X	X	X	X	X	X	X				X
Minimum Clock High	E	X	X	X	X	X	X	X	X	X	X	X				X
Minimum Period	E	X	X	X	X	X	X	X	X	X	X	X				X

Table 6-6 List of Patterns for Switching Speed Tests

and are also verified at the Y outputs. Pattern 1 is also used for several other switching speed tests as shown in Table 6-6. It is used with an early strobe (checking the outputs before the rising edge of the clock) and a late strobe (checking the outputs after the rising edge of the clock). For some tests the D input data is inverted. The output data for Pattern 1 is dependent on whether an early or late strobe is used and whether normal or inverted D input data is applied.

Pattern 2

This pattern loads data words into the Register/Counter to check the D set-up time to R and the $\overline{RLD}$ set-up time. The Register/Counter is loaded first with DW(J) by Instruction C (LDCT) and then again with $\overline{DW(J)}$ from the $\overline{RLD}$ input. The Register/Counter contents are verified for both data words and the subroutine is repeated for J = 1 - 20. Pattern 2 is also used to test many other switching speed parameters with early and late strobes and with normal and inverted D input data.

Pattern 3

The delay from CP to Y via the Register/Counter for Instruction 0 - 7 and A - E is checked by using the $\overline{RLD}$ input to load it first with $\overline{DW(J)}$ and then with DW(J) for J = 1 - 13. The Register/Counter contents are verified at the Y outputs. For Instructions 0 - 7 and A - E the delays from CP to Y via the Program Counter and Stack registers are checked with functional test patterns.

Pattern 4

Three different subroutines are contained in this pattern to verify the operation of the Zero Detector for the three instructions (8, 9, and F) which are dependent on its output. In each subroutine Instruction C is used to load data word "001" into the Register/Counter. Then the Register/Counter is decremented on Instruction 8, 9, or F and the PLA selects the proper Multiplexer output for the condition when the Register/Counter contents are zero. The other registers are loaded with DW(J), $\overline{DW(J)}$, or AAA. If the correct data source is not selected within the maximum delay, the incorrect data word will be read at the Y outputs. The subroutines are repeated for J = 1 - 20. This pattern verifies the delay (CP to Y when I = 8, 9, or F) for the Register/Counter to decrement, for the Zero Detector output to indicate zero, and for the Multiplexer to change data sources.

Pattern 4A

Vendor A developed this pattern to check the CP to Y delay when the Register/Counter changes from all zeroes to a non-zero data word. Vendor A stated that the counter is optimized to count down and the CP to Y delay is the longest when the Register is changing from a zero value to a non-zero value (counting up). Three different subroutines are again used, one for each of the instructions (8, 9 and F) that are dependent on the Register/Counter contents.

Zeroes are first loaded into the Register/Counter. Then Instruction C is used to load the Register/Counter with D(K) where D(K) is defined as follows:

K	D(K)
1	000000000010
2	00000 000100
3	000000001000
4	000000010000
5	000000100000
6	000000111110

Instruction 8, 9, or F follows Instruction C and the delay for the Zero Detector output to affect the PLA and, finally, the Multiplexer and Y outputs is checked. The Multiplexer inputs are either DW(J) or $\overline{DW(J)}$ for the PC, F, and D inputs and if the CP to Y delay is not met, the incorrect Y outputs will be obtained.

GE questioned some of the Y outputs in Pattern 4A. Vendor A reviewed the pattern and determined that the timing waveforms need to be modified. The $\overline{CC}$ and $\overline{CCEN}$ inputs were applied at the beginning of the cycle but the Instruction inputs were not applied until after the rising edge of the clock. This caused the wrong outputs to occur. Vendor A will provide a revised timing diagram for the slash sheet.

Pattern 5

The three Enable outputs are checked with this pattern. All 16 instructions are performed to verify the correct states on each output and to insure that the maximum propagation delays are not exceeded. No clock is used for this pattern.

General Tests and Comments

Most of the switching speed parameters were thoroughly checked by Vendor A's tests. The tester's XOR mode was used for the input-under-test. This mode forces the complement state on the input pin prior to the tests. The necessary transitions from 0 to 1 and 1 to 0 were made for each parameter to check the delay, set-up, and hold times. The only pattern that Vendor A did not provide was a High-Impedance pattern to check the delay from the $\overline{OE}$ input to the Y outputs. GE added a pattern which uses Instruction 2 to force the Y outputs to all logic "1" or logic "0" and enables the High-Impedance capability. This pattern was added to the functional and the switching speed tests.

The functional test and some of the switching speed test patterns are used to verify the minimum clock high and low times. A maximum frequency test was added by GE which uses the same patterns and specifies a limit obtained from Vendor A.

After the RESET operation (Instruction 0) is performed by some of the switching speed patterns, the Stack output is checked. It is normally undefined but the patterns specify that after RESET a certain output should be present. After discussion with Vendor A and review of the logic diagram, GE determined that the Stack output is predictable and is dependent on the previous Stack output, the Program Counter output, and the Stack Holding Register output.

SECTION VII

MIL-M-38510/440 UPDATE

The 2901A Four Bit Microprocessor Slice was characterized and the MIL-M-38510/440 slash sheet developed for it on a previous RADC contract as described in Report RADC-TR-78-138. As Vendor A reviewed the slash sheet and began writing test programs to conform to its requirements it became necessary to make changes to the specification.

This section describes the major changes made to the slash sheet since its original release. Since the 2901A is not qualified as yet, additional changes may be required in the future.

1. Pattern modification

   Vendor A requested changes to some of the functional and AC test patterns to facilitate testing. The following is a brief description of the changes that were made.

   a) The RAM shift portion of the functional test was modified so that the entire RAM is preloaded prior to performing a given shift function on all sixteen addresses. Previously each location was checked separately.

   b) A new pattern was developed to test the propagation delays into and out of the high impedance state.

   c) Pattern 1 was augmented so that it could be used to test all of the propagation delays from the A_3 - A_0 inputs. Previously Patterns 1 and 2 had to be used.

   d) New AC patterns were developed to check the minimum clock high time and the maximum frequency for shifting the Q register. These patterns were variations of the RAM shift and Q register shift portions of the functional test.

2. Waveform changes

Waveforms were added as required, for some of the new patterns. Some of the waveforms in the slash sheet were modified to make them compatible with Vendor A's test capability or for clarification.

3. Addition of subgroups 7 + and 8 +

The functional test was originally specified to be run at the maximum frequency possible taking into account the worst case set up times and propagation delays. This was done to allow as many of the internal paths as possible to be checked dynamically.

Vendor A requested the addition of a functional test which could be run with relaxed timing. Since the 2901A is a bipolar device, large amounts of current are drawn as the transistors are switching. This results in the generation of noise on tester ground and device outputs during test. By relaxing the timing for the functional test, the outputs can settle before they are strobed thus resulting in fewer good devices being rejected.

Subgroup 7 + and 8+ were created for the relaxed timing functional test. They will be used for interim electrical parameters, final electrical test parameters, and Group C and D end point electrical parameters as specified in Table II of the slash sheet. Subgroups 7 and 8 will still be used for final electrical test parameters and Group A test requirements as specified in Table II of the slash sheet.

4. Threshold test development

Functional tests are usually specified to be performed with V_{IH} and V_{IL} at their minimum and maximum values respectively. For the 2901A this would be V_{IH} = 2.0 V and V_{IL} = 0.8 V. However, because of the tester noise problem described above, Vendor A requested that the input levels be relaxed. It was agreed that V_{IH} = 2.4 V and V_{IL} = 0.5 V

could be used for the functional test input levels if it was verified during the DC tests that the device would operate at the specified input thresholds.

A sequence of vectors was developed for this purpose. All of the inputs are sensitized to outputs to guarantee that any failures will be detected. These vectors minimize the number of inputs switching at one time and they can be run at a slow rate to reduce the effects of tester noise.

5. Addition of double ended limits

Double ended limits were added for V_{OL}, V_{OH}, and propagation delays which have been specified with single ended limits in the past. The added limits will be used only during the qualification of a device. They were added to guarantee that all devices qualified to a slash sheet are similar in performance for a range of parameter values.

6. Miscellaneous slash sheet changes

Other changes to the slash sheet included eliminating all references to circuit B and specifying the operating region at $T_C = 125°C$. The latter change was required so that the maximum allowable junction temperature of the device would not be exceeded. This also eliminates the ambiguity of the definition of ambient temperature.

SECTION VIII

REVIEW OF MIL-M-38510/421

The 8212 eight bit I/O port was characterized and the MIL-M-38510/421 shash sheet developed for it on a previous RADC contract as described in Report RADC-TR-78-138. The functional test included in the slash sheet was developed by GE following a gate level analysis of the 8212.

Vendor C requested that they be allowed to qualify the 8212 using their in place test capability in lieu of the tests specified in the shash sheet. They specified their tests in a proposed Table III A which would be added to the slash sheet and could be used in place of Table III. Table III A was submitted to RADC and DESC along with a list of differences between it and Table III.

GE reviewed this information and contacted Vendor C for clarification. They provided additional information which answered some of the questions that existed. It was found that the submitted tests do not test all of the slash sheet parameters and that the information provided in Table III A does not provide sufficient and accurate conditioning for those that are done.

GE submitted a list of comments and recommendations regarding the proposed tests to RADC, DESC, and Vendor C. Vendor C is reviewing these recommendations so that a disposition of the 8212 testing can be made.

SECTION IX

PRELIMINARY REVIEW OF MEGATEST Q8000 TESTER APPLICABILITY TO MIL-M-38510 TESTING

As more and more vendors began to use dedicated benchtop testers in their test areas it became necessary for GE to learn the capabilities of these machines so that it could be determined whether they can be used for testing MIL-M-38510 devices. One of the more popular dedicated testers is the Megatest Q8000. It was the first one selected for evaluation because Vendor C is using it to test the 8048 which is being characterized on another contract (F30602-78-C-0189) with RADC. In addition, Vendor C has proposed using their existing Q8000 tests for the 8212 Eight Bit I/O Port and the 8224 Clock Generator and Driver in lieu of those specified in the slash sheets for these devices. The review of the alternate 8212 tests is described in Section VIII of this report.

Megatest was contacted for information on the Q8000. They provided a data package which included a sales brochure, a test language users guide, and hardware, operators, and programming and interface manuals. This information was reviewed and Megatest was contacted for clarification of some of the items.

The Q8000 is designed to perform functional tests and check all AC and DC parameters for any LSI device (up to 40 pins and in TTL, NMOS, or 5 V CMOS). The source of data for the functional tests is provided by a removable module called a Functional Data Module (FDM). Three basic types of FDM's are available. The universal ROM/PROM module can be used to test and program any ROM or EPROM up to 32 K bits; the truth table module is used for stored response testing of simple devices; and the reference system module is used to emulate complex LSI devices in real time. The reference system module contains a device identical to the one under test and several support chips. Inputs are applied to the reference device and then to the device under test, on the next clock cycle, after being conditioned by the pin electronics. The outputs of the device under test are compared to the outputs of the reference device which were stored on the previous cycle. The Q8000 has a single threshold comparator on each pin so two or more passes have to be made through the test if it is desired to compare the outputs at more than one level - eg. V_{OH}, V_{OL}, etc.

DC parameters such as power supply, input, and output currents and AC parameters such as propagation delays, setup and hold times, and clock cycle times can be checked. The setup and hold times and the output strobe times are controlled by one shots on the FDM.

Several potential problem areas relative to the use of a Q8000 need to be investigated before it is deemed acceptable for testing MIL-M-38510 devices. Some of these are described below:

1. Use of one shots - The use of one shots to control the timing for the test could result in a problem of parameter repeatability and might necessitate frequent calibration of the FDM.

2. FDM space problem - The FDM has space on it for 26 9602 one shots and approximately 100 16 pin IC's (less if larger IC's are used). If the FDM circuitry requires additional space then test compromises have to be made. This problem was encountered during the evaluation of Vendor C's test for the 8212 which is a relatively simple device. All of the AC parameters are not being checked because there is not enough space for all of the one shots required.

3. Pattern documentation - The use of a reference system module reduces program storage because a single command generates many bits of a test pattern. However, unless the test program is well documented it is not apparent what data is being operated on because it is obtained from a memory on the FDM and the outputs are not known since they exist only temporarily in the FDM. Knowledge of the circuitry on the FDM and the contents of any memories used is required to determine what the inputs and outputs are at any given time. A listing of the entire test pattern should be available.

These are some of the problems encountered to date in evaluating the Q8000. Others may arise as the characterization of the 8048 continues. The final report for contract F30602-78-C-0189 will include a continuation of GE's evaluation of the Q8000.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.